Online Scheduling of a Residential Microgrid via Monte-Carlo Tree Search and a Learned Model

Hang Shuai , *Member, IEEE*, and Haibo He , *Fellow, IEEE*

Abstract—The uncertainty of distributed renewable energy brings significant challenges to economic operation of microgrids. Conventional online optimization approaches require a forecast model. However, accurately forecasting the renewable power generations is still a tough task. To achieve online scheduling of a residential microgrid (RM) that does not need a forecast model to predict the future PV/wind and load power sequences, this article investigates the usage of reinforcement learning (RL) approach to tackle this challenge. Specifically, based on the recent development of Model-Based Reinforcement Learning, *MuZero* (Schrittwieser *et al.*, 2019) we investigate its application to the RM scheduling problem. To accommodate the characteristics of the RM scheduling application, an optimization framework that combines the model-based RL agent with the mathematical optimization technique is designed, and long short-term memory (LSTM) units are adopted to extract features from the past renewable generation and load sequences. At each time step, the optimal decision is obtained by conducting Monte-Carlo tree search (MCTS) with a learned model and solving an optimal power flow sub-problem. In this way, this approach can sequentially make operational decisions online without relying on a forecast model. The numerical simulation results demonstrate the effectiveness of the proposed algorithm.

Index Terms—Deep reinforcement learning, Monte-Carlo tree search (MCTS), microgrid, online optimization.

Nomenclature

Superscript, Subscript, Indices and Sets

$b, grid$	Battery and utility grid.
ch, dis	Charge and discharge mode of battery.
$g, \mathscr{G}$	Index and set of dispatchable generators.
K, k	Number and index of hypothetical time steps.
L	Load power demand.
T, t, Γ	Number, index, and set of time steps.
$i, j, \mathscr{N}$	Index and set of buses.
Υ	Set of branches in microgrid.
max, min	Maximum and minimum value.
wt, pv	Wind turbine and PV panel.

Variables

P, Q	Active power and reactive power.
SoC	State of charge of battery.
l_{ij}, v_i	Square of branch current and bus voltage.
I^{ch}, I^{dis}	Charge and discharge state of battery.
s, x, r	State, decision, and reward of microgrid.
$\hat{s}, \hat{x}, \hat{r}$	Internal state, decision, and reward of microgrid.

Parameters

p_{cur}	Renewable energy curtailment cost coefficient.
r_{ij}, x_{ij}	Resistance and reactance of power line.
α_g, β_g, c_g	Fuel cost coefficient of dispatchable DGs.
ρ	Unit degradation price of battery.
η	Efficiency of battery.
E^{min}, E^{max}	Minimum and maximum energy in battery.
γ	Discout factor.
Δt	Time resolution.

Functions

$C(\cdot), r(\cdot)$	Cost function and reward function.
$h_\theta(\cdot), g_\vartheta(\cdot)$	Representation and dynamic network.
$f_\phi(\cdot)$	Prediction network.
$l(\cdot)$	Loss function.
$Q(\hat{s}, \hat{x})$	Mean value function.
$P(\hat{s}, \hat{x})$	Policy function.
$S(\hat{s}, \hat{x})$	Transition function.

I. Introduction

MICROGRID is a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid [2]. It is becoming a widely adopted technology to utilize distributed energy resources (DERs) as their capability of reducing greenhouse emissions, improving consumers' supply reliability, enhancing power grid resiliency, etc., [3], [4]. For instance, there have been 6610 microgrid projects representing 31.7 GW of planned and installed power capacity globally [5], as of March 2020. However, the intermittent and uncertainty of integrated renewable energy bring significant challenges to the reliable and economic operation of microgrids. The optimal optimization and control strategies are the key techniques to ensure the economic operation of microgrids. As a result, the optimization of microgrids has obtained extensive research, and a variety of microgrid energy

management algorithms (see [6] and the references therein) have been proposed, such as, linear and nonlinear programming methods, dynamic programming and rule-based methods, meta-heuristic approaches (particle swarm optimization, genetic algorithm, etc.), artificial intelligence methods (fuzzy logic, neural network, multi-agent system, etc.), stochastic programming, and robust optimization [7]. Besides, researchers also proposed several other optimization methods [6], [8]. For instance, a novel peer-to-peer optimization method was proposed in [8], which greatly improved the renewables consumption by incentivizing microgrids to trade energy.

However, the above microgrid optimization methods are mainly proposed to solve the microgrid planning problems (see [7], [9], [10]) or day-ahead scheduling problems (see [11], [12]). Using these methods, we can make day-ahead scheduling according to the forecast information of all the future system state and the statistic distribution information of the uncertainties in the system. Since the prediction errors come from both the power generation side and the demand side, the actual operational decisions are re-optimized sequentially in the intra-day online optimization process according to the updated short-term forecasting information provided by the forecast model of the system. Model predictive control (MPC) is a traditional online optimization method that has been applied in microgrids [13], [14]. But, the performance of the MPC approach depends on the precision of the forecasting generation/load power and the fitted statistic distribution information. To reduce the influence of renewable energy forecast errors on the economic operation of microgrids, Shuai *et al.* [15] proposed a cost function approximation (CFA) based online optimization algorithm. However, the CFA based online optimization strategy is still rely on the short-term renewable energy forecast information.

To obtain optimal online operation decisions, researchers have made some efforts and proposed several optimization approaches to reduce the dependence on forecasting information provided by forecast models of microgrids. A class of heuristic online algorithms, called CHASE [16], *h*CHASE [17], [18], have been proposed recently. The CHASE algorithm can achieve theoretical performance guarantee without any future information. However, the CHASE algorithm is designed to solve single or multiple homogeneous local generators. Then, a more general retrospection-inspired online scheduling algorithm, *h*CHASE, is proposed. Shi *et al.* [19] proposed a Lyapunov optimization based online optimization strategy for microgrids, and simulations demonstrate that the algorithm can make sub-optimal decisions without any prior statistical knowledge of the stochastic processes. Ma *et al.* [20] proposed an online alternating direction method of multipliers (ADMM) based distributed algorithm for online optimization of microgrids. The online ADMM algorithm proposed in [20] does not require any forecast data to proceed, which avoids problems caused by inaccurate forecasting. However, simulation results in [20] indicate that there exists constraint violations. More concretely, the online ADMM algorithm cannot ensure the active/reactive power constraints and the voltage constraints be fulfilled at each time period.

Li *et al.* [21] proposed an online learning-aided energy management algorithm and combined it with ADMM algorithm to facilitate the real-time implementation. Rahbar *et al.* [22] developed a dynamic programming (DP) based sequential online optimization algorithm. The authors of this article proposed an approximate dynamic programming (ADP) based microgrid online optimization approaches [23], [24]. After trained off-line using the day-ahead forecasting information, the ADP algorithm can obtain near-optimal online decisions only according to the current system state and the well-trained value functions.

Although the above online optimization approaches reduced the dependence on intra-day forecasting information, the historical renewable and load power data are not been fully utilized in the optimization process. With the rapid development of machine learning techniques, researchers have made some efforts to apply deep learning and model-free reinfrocement learning algorithms to solve microgrid energy management problem [25]. Besides, prior research works [26], [27] indicate that the intelligent agent trained based on historical data is more general to adapt to the unknown situation in the future. Thus, in order to learn to operate microgrids from historical data, researchers proposed the model-free deep reinforcement learning (DRL) based online optimization algorithms [27]–[29] recently. In [27], [29], the historical data are used to train the designed Deep Q Network (DQN) and Deep Deterministic Policy Gradient (DDPG) algorithms to achieve a good online decision performance.

In this work, we investigate the application of a model-based deep reinforcement learning (MB-DRL) algorithm developed by reference [1] to solve the microgrid scheduling problem. Different from the *AlphaGo* [30] and *AlphaGo Zero* [31] algorithms that need to know the dynamics of the environment, the MB-DRL algorithm, called *MuZero*, that proposed in [1] is a more general and more powerful reinforcement learning algorithm that can achieve superhuman performance in a range of challenging Atari games. Specifically, the *MuZero* algorithm combines a tree-based search policy with a learned model that consists of three networks (representation network, dynamic network, and prediction network), and can make decisions without any knowledge of the underlying dynamics of the environment. It is worth noting that there are many differences between the microgrid optimization problem and playing Atari games. For example, there are plenty of equality and inequality constraints in microgrids, and the action space of the optimization problem in this work is huge because the decision variables are multidimensional. These differences bring significant challenges to the application of the algorithm. This motivated us to explore the application potential of the MB-DRL algorithm in solving microgrid online optimization problems.

This work focuses on the online optimization of a residential microgrid (RM) under uncertainty. The optimization problem in this article is formulated as a mixed integer second-order cone programming (MISOCP) problem. To solve this problem, we reformulate it as a Markov Decision Process (MDP), and a MB-DRL based RM scheduling algorithm is designed. The advantage of the developed algorithm is that

it can make online decisions sequentially without relying on the renewable and load power prediction from forecast models. The main contributions of this work are summarized as follows:

1) An MB-DRL based RM optimization approach is developed based on *MuZero* [1] algorithm, with the aim of conceiving an online optimization agent that can teach itself how to optimally operate the RM system through self-play.

2) To deal with the application challenges brought by a large number of operational constraints and the huge decision space, a framework that combines the MB-DRL algorithm with mathematical optimization techniques is designed.

3) A new representation network architecture is designed. Different from most Atari games existing large spatial resolution in observations, the observations of microgrids have a strong time correlation. Thus, for the representation network in the proposed scheduling algorithm, we first utilize three LSTM units to extract features from past solar power, wind power, and load power, respectively. Then, the feature vectors are concatenated with the current state information and fed into the input layer of a fully connected neural network.

4) Simulation results demonstrate the developed MB-DRL approach can learn to solve the complex RM optimization problem by self-play, and can obtain better online optimization performance than many state-of-the-art online optimization algorithms.

The rest of this article is organized as follows. The online scheduling of the residential microgrid is formulated in Section II. Section III presents the MB-DRL based online optimization algorithm. The simulation results are given in Section IV. Section V concludes the paper.

II. SCHEDULING MODEL OF THE RESIDENTIAL MICROGRID

The microgrid system investigated in this article is a RM system with a high penetration of renewable energy, which consists of PV-based DER units, wind turbine based DER units, diesel engine generator, energy storage device, electrical loads, and a smart energy management unit. The microgrid is connected with the utility grid, so it can purchase power from the utility grid when the RM suffers from power supply shortage. On the other hand, the microgrid can also sell its surplus energy to the utility grid. As the uncertainties from both renewable energy resources and the demand side, an energy management system (EMS) is needed to coordinate all the generation and load resources (including the energy storage system) to ensure the secure and economic operation of the system. So, one of the functions of smart EMS unit is to make online energy scheduling decisions according to the actual load demand and the available generation capacity from all power sources. In this article, we investigate the designation of a smart online optimization algorithm to achieve the optimal operation of the system. In the following section, we formulate the optimization model of the RM system as a MISOCP problem.

From the perspective of a system operator, the objective of the online energy management is to make decisions at each time step in order to minimize the operational cost under the uncertainties from renewable energy and electricity load, which leads to the following economic dispatch model:

$$\min \quad \Xi\left\{\sum_{t=\Delta t}^{T}\left(\sum_{g\in\mathscr{G}}C_g^{DG}\left(P_g^{DG}(t)\right) + C_{grid}\left(P_{buy}^{grid}(t), P_{sell}^{grid}(t)\right)\right.\right.$$
$$\left.\left. + C_{bat}\left(P^b(t)\right) + C_{cur}\left(P_{cur}^{ren}(t)\right)\right)\right\} \quad (1)$$

$$C_g^{DG}\left(P_g^{DG}(t)\right) = \left(\alpha_g\left(P_g^{DG}(t)\right)^2 + \beta_g P_g^{DG}(t) + c_g\right)\Delta t \quad (2)$$

$$C_{grid}\left(P_{buy}^{grid}(t), P_{sell}^{grid}(t)\right) = \left(p_{buy}(t)P_{buy}^{grid}(t) - p_{sell}(t)P_{sell}^{grid}(t)\right)\Delta t \quad (3)$$

$$C_{bat}\left(P^b(t)\right) = \rho|SoC(t) - SoC(t - \Delta t)| \quad (4)$$

$$C_{cur}\left(P_{cur}^{ren}(t)\right) = p_{cur}\left(\bar{P}^{pv}(t) + \bar{P}^{wt}(t) - P^{pv}(t) - P^{wt}(t)\right)\Delta t \quad (5)$$

where $\Xi\{\cdot\}$ represents the expectational operator. The first term in (1) represents the fuel cost of all dispatchable generators during a single time period, which is a quadratic function of the active power generation $P_g^{DG}(t)$, as shown in (2). The second term of the objective function is the power exchange cost which can be calculated as shown in (3), where $P_{buy}^{grid}(t)$ and $P_{sell}^{grid}(t)$ are respectively power purchased and sold by the RM, and $p_{buy}(t)$ and $p_{sell}(t)$ are energy purchase and sell price, respectively. Note that $p_{sell}(t)$ is usually lower than or equal to $p_{buy}(t)$, which is determined by electricity price policies in different states. The third term in (1) denotes the degradation cost of the battery system. The degradation cost caused by charging and discharging can be linearly approximated by the change between two consecutive SoC [25], [32], as shown in (4). The last term is the renewable energy curtailment cost of the system, where $P^{pv}(t)$ and $P^{wt}(t)$ are respectively the dispatched power of PV energy and wind energy. $\bar{P}^{pv}(t)$ and $\bar{P}^{wt}(t)$ are respectively the maximum available PV power and wind power currently. In the above equations, t is the time period index.

The RM system also needs to satisfy the following operational constraints:

$$P_g^{DG,min} \leq P_g^{DG}(t) \leq P_g^{DG,max}, \forall t \in \Gamma, \forall g \in \mathscr{G} \quad (6)$$

$$\left(P_g^{DG}(t)\right)^2 + \left(Q_g^{DG}(t)\right)^2 \leq \left(S_g^{DG,max}\right)^2, \forall t \in \Gamma, \forall g \in \mathscr{G} \quad (7)$$

$$0 \leq P^{wt}(t) \leq \bar{P}^{wt}(t), \forall t \in \Gamma \quad (8)$$

$$\left(P^{wt}(t)\right)^2 + \left(Q^{wt}(t)\right)^2 \leq \left(S^{wt,max}\right)^2, \forall t \in \Gamma \quad (9)$$

$$0 \leq P^{pv}(t) \leq \bar{P}^{pv}(t), \forall t \in \Gamma \quad (10)$$

$$\left(P^{pv}(t)\right)^2 + \left(Q^{pv}(t)\right)^2 \leq \left(S^{pv,max}\right)^2, \forall t \in \Gamma \quad (11)$$

$$\begin{cases} 0 \leq P_{buy}^{grid}(t) \leq P_{buy}^{grid,max} \\ 0 \leq P_{sell}^{grid}(t) \leq P_{sell}^{grid,max} \end{cases} \forall t \in \Gamma \quad (12)$$

$$0 \leq Q^{grid}(t) \leq Q^{grid,max}, \forall t \in \Gamma \quad (13)$$

$$\begin{cases} 0 \leq P^{ch}(t) \leq I^{ch}(t)P^{ch,max} \\ 0 \leq P^{dis}(t) \leq I^{dis}(t)P^{dis,max} \end{cases} \forall t \in \Gamma \quad (14)$$

$$P^b(t) = I^{dis}(t)P^{dis}(t) - I^{ch}(t)P^{ch}(t), \forall t \in \Gamma \quad (15)$$

$$I^{dis}(t) + I^{ch}(t) \leq 1, \forall t \in \Gamma, \{I^{dis}(t), I^{ch}(t)\} \in \{0, 1\} \tag{16}$$

$$\left(P^b(t)\right)^2 + \left(Q^b(t)\right)^2 \leq \left(S^{b,max}\right)^2, \forall t \in \Gamma \tag{17}$$

$$SoC(t + \Delta t) = SoC(t) + \eta^{ch}\frac{P^{ch}(t)}{E^{max}}\Delta t - \frac{1}{\eta^{dis}}\frac{P^{dis}(t)}{E^{max}}\Delta t \tag{18}$$

$$SoC^{min} \leq SoC(t) \leq SoC^{max}, \forall t \in \Gamma \tag{19}$$

$$\begin{cases} P_j(t) = P_{ij}(t) - r_{ij}l_{ij}(t) - \sum_{m:(j,m)\in\Upsilon} P_{jm}(t) \\ Q_j(t) = Q_{ij}(t) - x_{ij}l_{ij}(t) - \sum_{m:(j,m)\in\Upsilon} Q_{jm}(t) \end{cases} \forall (i,j) \in \Upsilon, \forall t \in \Gamma \tag{20}$$

$$v_j(t) = v_i(t) - 2\left(r_{ij}P_{ij}(t) + x_{ij}Q_{ij}(t)\right) + \left(r_{ij}^2 + x_{ij}^2\right)l_{ij}(t), \forall t \in \Gamma \tag{21}$$

$$V_i^{min} \leq | V_i(t) | \leq V_i^{max}, \forall t \in \Gamma, \forall i \in \mathcal{N} \tag{22}$$

$$l_{ij}(t) = \frac{P_{ij}(t)^2 + Q_{ij}(t)^2}{v_i(t)}, \forall (i,j) \in \Upsilon, \forall t \in \Gamma \tag{23}$$

Constraints (6)-(7) are the generator capacity constraints of controllable DGs, and (7) ensures that the reactive power generation of controllable DGs will not exceed its capacity; constraints (8)-(11) are the generator capacity constraint of renewable generators, and the dispachable active generation power of the PV and wind turbine are limited by the current available renewable power $\bar{P}^{wt}(t)$ and $\bar{P}^{pv}(t)$, as shown in (8) and (10); constraints (12)-(13) are the power exchange limitation between the RM and utility grid; constraints (14)-(19) are the battery storage related constraints, and constraint (14) ensures the active charge/discharge power will not exceed power limitations of the battery; constraint (16) avoids simultaneous charging and discharging of batteries; constraint (17) bounds the complex power of the battery, where $S^{b,max}$ is the capacity of the battery inverter; constraint (18) is the energy transition constraint of the battery.

Constraints (20)-(23) are the power flow model of the RM, where $P_{ij}(t)$ and $Q_{ij}(t)$ are the complex power flowing from bus i to j. In this work, the branch flow model [33] is adopted to model the steady-state power flows in the power network, where $(i,j) \in \Upsilon$ denotes the branch between bus i and j, and $r_{ij} + \mathbf{i}x_{ij}$ is the complex impedance of the branch. In (20), the complex net load of each bus i is represented by $P_i(t) + \mathbf{i}Q_i(t)$, which is the load power $P_i^L(t) + \mathbf{i}Q_i^L(t)$ minus the generation power. In the power flow model $l_{ij}(t) = | I_{ij}(t) |^2$ and $v_i(t) = | V_i(t) |^2$, where $I_{ij}(t)$ and $V_i(t)$ are the branch current and complex voltage at each bus, respectively. The magnitude of bus voltage is limited by (22). To formulate the optimization problem as a convex problem, we relax the quadratic equality constraint (23) to the following inequality

$$l_{ij}(t) \geq \frac{P_{ij}(t)^2 + Q_{ij}(t)^2}{v_i(t)}, \forall (i,j) \in \Upsilon, \forall t \in \Gamma \tag{24}$$

Finally, we can make the optimal decisions by solving the following MISOCP problem:

$$\min_{x_t} \Xi\left\{\sum_{t=\Delta t}^{T}\left(\sum_{g\in\mathcal{G}} C_g^{DG}\left(P_g^{DG}(t)\right) + C_{grid}\left(P_{buy}^{grid}(t), P_{sell}^{grid}(t)\right)\right.\right.$$
$$\left.\left. + C_{bat}\left(P^b(t)\right) + C_{cur}\left(P_{cur}^{ren}(t)\right)\right)\right\} \tag{25}$$

s.t. $(6) - (22), and\ (24)$

where x_t is the decision vector at time t, and

$$x_t = \left(S_g^{DG}(t), S^b(t), S^{pv}(t), S^{wt}(t), P_{buy}^{grid}(t), P_{sell}^{grid}(t), Q^{grid}(t),\right.$$
$$\left. P_{ij}(t), Q_{ij}(t), v_i(t), l_{ij}(t)\right) \tag{26}$$

In (26), $S_g^{DG}(t)$, $S^{pv}(t)$, and $S^{wt}(t)$ represent the complex power generation of controllable DGs, PV panels, and wind turbines, respectively. $S^b(t)$ is the complex output power of the battery, and $S^b(t) = P^b(t) + \mathbf{i}Q^b(t)$.

In this article, we investigate the online optimization of the RM system, which means system operators need to make operational decisions according to the current state information of the RM. In the following section, we will introduce a data-driven method, namely the MB-DRL algorithm, to solve the online optimization problem.

III. MICROGRID ONLINE SCHEDULING VIA MCTS AND A LEARNED MODEL

In this section, a model-based deep reinforcement learning based RM online optimization approach is designed to optimally operate the RM under uncertainties from renewable energy and demand side. We first reformulate the above RM optimization problem as a MDP problem. Then, the adopted MB-DRL method is introduced. Finally, a specific online optimization algorithm for the RM is designed.

A. Problem Reformulation

To facilitate the application of reinforcement learning approach, the optimization problem shown in (25) is reformulated as a MDP problem. A MDP model includes some basic elements, namely state variables s_t, decision variables x_t, transition function, and reward function $r_t(\cdot)$, which will be defined in the following section. Note that different from the variable representation rules in the previous equations, we will subscript t to represent the variable at time t in the following sections, which is to be consistent with the variable naming convention in the field of reinforcement learning. For instance, the state variable is represented by s_t rather than $s(t)$. The state variables of the RM consist of the SoC of the battery, the active and reactive power demand of each bus $P_i^L(t)$ and $Q_i^L(t)$, the available PV generation $\bar{P}^{pv}(t)$, the available wind power generation $\bar{P}^{wt}(t)$, and the electricity price $p_{buy}^{grid}(t)$ and $p_{sell}^{grid}(t)$. The state variables of the RM at time t are defined below:

$$s_t = \left\{SoC(t), P_i^L(t), Q_i^L(t), \bar{P}^{pv}(t), \bar{P}^{wt}(t), p_{buy}^{grid}(t), p_{sell}^{grid}(t)\right\} \tag{27}$$

The decision variables of the RM at time t is given in (26). The SoC transition function has been given in (18). The objective of the online optimization is to minimize the operational cost of the RM, which is equivalent to maximize the total rewards of the system. So, we define the following reward function according to the objective
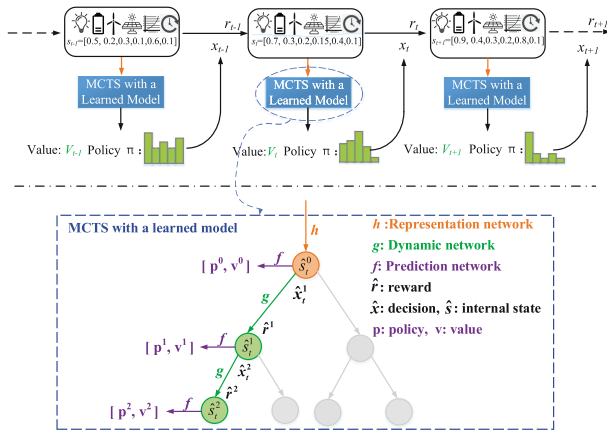
Fig. 1.   The architecture of the MCTS with a learned model [1].

function shown in (1):

$$r_t(s_t, x_t) = -\left(\sum_{g \in \mathcal{G}} C_g^{DG}\left(P_g^{DG}(t)\right) + C_{grid}\left(P_{buy}^{grid}(t), P_{sell}^{grid}(t)\right)\right.$$

$$\left. + C_{bat}\left(P^b(t)\right) + C_{cur}\left(P_{cur}^{ren}(t)\right)\right) \qquad (28)$$

where, $r_t(s_t, x_t)$ is the reward of taking decision x_t when RM in state s_t.

After reformulated the problem as a MDP problem, the next step is to design a reinforcement learning algorithm to make optimal decisions at each time step in order to maximize the cumulative rewards of the RM.

B. The Model-Based Deep Reinforcement Learning Method

In this work, we adopted the MB-DRL optimization approach proposed in [1]. In this section, the principle of the the adopted MB-DRL approach is briefly introduced.

The MB-DRL algorithm proposed in [1], called *MuZero*, combines Monte-Carlo Tree Search (MCTS) planning method with a learned neural network model. Similar to traditional MCTS, the *MuZero* algorithm involves iteratively building a search tree until some predefined computational budget (like a maximum iteration constraint) is reached, then the search is halted and the best action is determined according to the visit count of each action from the root node [34].

The difference between the two algorithms is that a learned model is introduced in *MuZero* to improve the performance of MCTS. Specifically, three different neural networks, which constitute the learned model, are introduced in the tree search process, as shown in Fig. 1. The first neural network is the representation network h_θ, which is used to encode past system states (or observations) until the current time-step t, $(s_1, \ldots, s_t)$, into an internal state $\hat{s}_t^0$. The second one is the dynamic network g_ϑ, which is used to get the next internal state $\hat{s}_t^k$ and the immediate reward $\hat{r}^k$ after taken an action $\hat{x}_t^k$ from an internal state $\hat{s}_t^{k-1}$. The last neural network is the prediction network, which is used to get the control policy $\mathbf{p}^k$

and value functions v^k according to the internal state $\hat{s}_t^k$.

$$\hat{s}_t^0 = h_\theta(s_1, \ldots, s_t) \qquad (29)$$

$$\hat{r}^k, \hat{s}_t^k = g_\vartheta\left(\hat{s}_t^{k-1}, \hat{x}_t^k\right) \qquad (30)$$

$$\mathbf{p}^k, v^k = f_\phi\left(\hat{s}_t^k\right) \qquad (31)$$

In (29)-(31), θ is the weights of the representation network; ϑ is ϑ of the dynamic network; ϕ represents the weights of the prediction network. The superscript k represents the variables of the kth hypothetical time-step. For instance, $\hat{s}_t^k$ denotes the internal state at the kth hypothetical time-step during the tree search conducted at the actual time-step t. It is worth noting that the internal state has no semantics of the environment state attached to it [1]. The purpose of the internal state is to improve the prediction accuracy of the above control policies $\mathbf{p}$, values v, and immediate rewards $\hat{r}$ that used in the tree search process. We can also find that the dynamic network is actually an environment model approximator, which can directly compute the next internal state according to the current state and the executed action. This makes the *MuZero* algorithm belongs to a model based reinforcement learning algorithm.

The three neural networks are trained off-line using the data generated by the self-play process of the *MuZero* algorithm. And after the neural network model has been well-trained, the optimal decisions of each time-step can be made forward through time according to the current state of the environment, as shown in Fig. 1. At each time-step t, an MCTS is performed to build a search tree. Note that an MCTS consists of a predefined number of simulations, and each simulation includes three steps, namely selection, expansion, and back-propagation. The above learned model is utilized in each simulation to help us to build the search tree. Then the best decision is sampled according to the visit count of the child nodes of the root node. The decisions are applied to the environment, and the agent gets a new observation of system state s_{t+1} and the actual reward r_{t+1} from the environment. The above procedure repeats until the end of the game. Here we just provided a brief description of the MuZero algorithm, and refer the readers to reference [1] for more details.

C. Proposed MB-DRL based Online Scheduling Algorithm for the Residential Microgrid

Originally, the *MuZero* algorithm is proposed to play games like Atari games and Go. In these games, the action is usually a single-dimensional variable and discrete values. So the action space at a single time-step is relatively small. For example, in Atari 2600 game Breakout, the action space is ['noop', 'fire', 'right', 'left', 'right fire', 'left fire']. However, the actions of the optimization problem in this article is a multidimensional continuous vector, as shown in (26). Thus the action space is huge. This makes us face the 'curse of dimensionality' when applying reinforcement learning method to solve our problem. Besides, there are plenty of equality and inequality constraints in the above RM model, which is difficult to directly handle for the *MuZero* algorithm. How to ensure the decisions

given by the reinforcement learning agent fulfill the complex constraints (6)-(22), and (24) is also a big challenge we face.

To solve the challenges, we propose to divide the decision variables shown in (26) into two categories. The first category contains the active charge/discharge decision of the battery $P^b(t)$, which is determined directly by the model based reinforcement learning algorithm. And we discretize the charge/discharge decision to facilitate the application of MCTS algorithm. Researchers have proposed a MCTS algorithm that can deal with continuous actions [35]. The extension, however, is left for future work. The second category includes the remaining decisions in (26), which are optimized as a single time period optimal power flow (OPF) sub-problem, as shown in (32).

$$\min_{x_t^s} \left(\sum_{g \in \mathcal{G}} C_g^{DG}\left(P_g^{DG}(t)\right) + C_{grid}\left(P_{buy}^{grid}(t), P_{sell}^{grid}(t)\right) \right.$$
$$\left. + C_{bat}\left(P^{b,*}(t)\right) + C_{cur}\left(P_{cur}^{ren}(t)\right) \right)$$
$$\text{s.t. } (6) - (13), (17), (20) - (22), \text{ and } (24) \qquad (32)$$

where, x_t^s denotes the decisions optimized in the OPF sub-problem. $P^{b,*}(t)$ is the optimal decision directly given by the MB-DRL method. The sub-problem (32) can be solve by second-order cone programming (SOCP) technique. Note that the active charge/discharge decision is fixed when solving the OPF sub-problem. The proposed optimization framework that combines the MCTS with the mathematical optimization technique will bring two advantages. The first advantage is that the action space handled by the MCTS is largely reduced. The second one is that most constraints can be handled by OPF sub-problem, which avoids letting the MCTS and the neural network model directly deal with a large number of constraints and this will greatly help the learning process of the agent. The schematic diagram of the proposed MB-DRL based online optimization algorithm for RM is shown in Fig. 2. In the figure, the charge/discharge decision $P^b(t)$ given by the MCTS algorithm is checked by the overcharge/overdischarge limitation unit to ensure the constraint (19) is fulfilled, as shown in (33).

$$P^{b,*}(t) = \begin{cases} P^b(t), & SoC^{min} \le SoC(t+\Delta t) \le SoC^{max} \\ -\frac{E^{max} \cdot (SoC^{max} - SoC(t))}{\eta^{ch}\Delta t}, & SoC(t+\Delta t) > SoC^{max} \\ \frac{(SoC(t)-SoC^{min}) \cdot E^{max} \cdot \eta^{dis}}{\Delta t}, & SoC(t+\Delta t) < SoC^{min} \end{cases}$$
$$(33)$$

where, $SoC(t+\Delta t)$ can be computed using (18) according to $P^b(t)$.

The multi-time period optimization problem shown in (25) is decomposed to multiple single time period sub-problems in this work according to Bellman's optimality principle, and the optimal decisions of each time-step are optimized using MCTS planning method combined with the learned neural network model, as shown in **Algorithm 1**. We assume all the states of the RM are observable. At time-step t, the states of the RM are collected, and are fed to the representation network to compute the internal state of the system using (29). Then, the
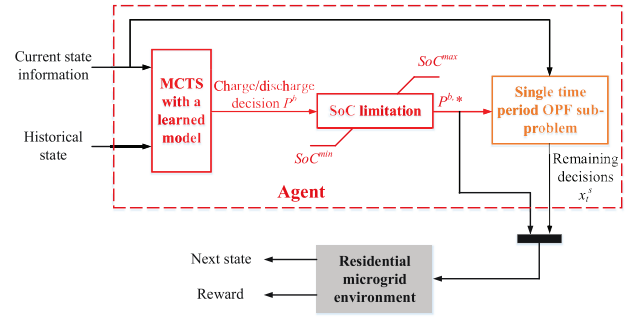

Fig. 2. The online optimization process at each time-step.

Algorithm 1 The Proposed MB-DRL Based Online Optimization Algorithm

---

1: Load the pre-trained neural network model which includes the representation network, the dynamic network, and the prediction network; load the RM system parameters.
2: **for** $t = \Delta t, 2\Delta t, \ldots, T$ **do**:
3:    Get the current state information of the system.   ▷ (27).
4:    Compute the internal state of the RM according to the current state information and the historical state information. ▷ (29).
5:    Create the root node according to the computed internal state, and set $n = 1$.
6:    **while** $n \le N$ **do**
7:       Perform the selection step start from root node. ▷ (34).
8:       Perform the expansion step.   ▷ (35)-(37).
9:       Perform the backpropagation step.   ▷ (38)-(40).
10:       $n = n + 1$.
11:    **end while**
12:    Get the optimal charge/discharge decision $P^{b,*}(t)$ that correspond to the most visited child nodes of the root node.
13:    Overcharge/overdischarge check and get the optimal decision $P^{b,*}(t)$.   ▷ (33).
14:    Fix the charge/discharge decision of the battery to be $P^{b,*}(t)$ and solve the OPF sub-problem (32) to compute the optimal decisions $x_t^{s,*}$.
15:    Execute the optimal decisions and calculate the next state of the RM system.
16: **end for**

---

root node is created according to the obtained internal state $\hat{s}_t^0$. Starting from the root node, a search tree is constructed by iteratively performing N simulations (see lines 6 to 11 of **Algorithm 1**). And the optimal output power of the battery can be determined according to the visit frequency of the child nodes of the root node. Then use (33) to make sure the SoC limitation will not be violated. Finally, the remaining decisions x_t^s are determined by solving the OPF sub-problem, and the optimal decisions are executed. In the next time-step, the above optimization process is repeated.

We can find that the key step is the search tree building process (see lines 4 to 11 of **Algorithm 1**). The tree consists of nodes and edges. Each node is associated with an internal state $\hat{s}$. For each decision $\hat{x}$ from $\hat{s}$ there is an edge $(\hat{s}, \hat{x})$ that leads to a child node. The information stored in the edge $(\hat{s}, \hat{x})$ is $\omega = \{N(\hat{s}, \hat{x}), Q(\hat{s}, \hat{x}), P(\hat{s}, \hat{x}), R(\hat{s}, \hat{x}), S(\hat{s}, \hat{x})\}$. N, Q, P, R, and S represent visit counts, mean value, policy, reward, and state transition, respectively. For simplicity, $\hat{s}_t^k$ and $\hat{x}_t^k$ will be represented by $\hat{s}^k$ and $\hat{x}^k$ in the following context. Each simulation includes the following three steps.

1) Selection: Starting from the root state $\hat{s}^0$, the simulation selects a decision and reaches to the corresponding child node. The Selection step finishes when the simulation arrives at a leaf node $\hat{s}^l$. For each hypothetical time-step $k = 1, 2, \ldots, l$ of the simulation, a decision $\hat{x}^k$ is selected using (34).

$$\hat{x}^k = \arg\max_{\hat{x}} \left[Q(\hat{s}, \hat{x}) + P(\hat{s}, \hat{x}) \cdot \frac{\sqrt{\sum_\varsigma N(\hat{s}, \varsigma)}}{1 + N(\hat{s}, \hat{x})} \right.$$
$$\left. \times \left(c_1 + \log\left(\frac{\sqrt{\sum_\varsigma N(\hat{s}, \varsigma)} + c_2 + 1}{c_2} \right) \right) \right] \quad (34)$$

where, $Q(\hat{s}, \hat{x})$ is the value that takes decision $\hat{x}$ from state $\hat{s}$, which represents the average reward for taking this decision (exploitation term). The second part of the right-hand side of (34) is the exploration term which can encourage the simulation to take decisions that have been less selected. $P(\hat{s}, \hat{x})$ is the prior probability of taking decision $\hat{x}$ from state $\hat{s}$. c_1 and c_2 are the constant value which is used to balance the exploitation and exploration term. In general, we can set $c_1 = 1.25$ and $c_2 = 19652$, as suggested by reference [1]. For $k < l$, the next state and reward are obtained by looking up the state transition table $\hat{s}^k = S(\hat{s}^{k-1}, \hat{x}^k)$ and corresponding reward table $r^k = R(\hat{s}^{k-1}, \hat{x}^k)$, respectively.

2) Expansion: When the simulation steps to the leaf node of the tree at the final time-step l, a new child node will be added to the tree. The state of the new node $\hat{s}^l$ is computed by the learned dynamic network as shown in (35). Besides, the corresponding reward $\hat{r}^l$ is also computed, and the obtained new internal state and reward will be stored in the corresponding tables, $R(\hat{s}^{l-1}, \hat{x}^l) = \hat{r}^l$, $S(\hat{s}^{l-1}, \hat{x}^l) = \hat{s}^l$.

$$\hat{r}^l, \hat{s}^l = g_\vartheta\left(\hat{s}^{l-1}, \hat{x}^l\right) \quad (35)$$

The policy and value correspond to state $\hat{s}^l$ are computed using the learned prediction network:

$$\mathbf{p}^l, v^l = f_\theta\left(\hat{s}^l\right) \quad (36)$$

Then, the information stored in the new edge $(\hat{s}^l, \hat{x})$ is initialized to:

$$\omega = \left\{ N\left(\hat{s}^l, \hat{x}\right) = 0, Q\left(\hat{s}^l, \hat{x}\right) = 0, P\left(\hat{s}^l, \hat{x}\right) = \mathbf{p}^l \right\}. \quad (37)$$

3) Backpropagation: At the end of the simulation, the information of the new leaf node is backpropagated along the trajectory to update the statistics of all the edges in the simulation path. For $k = l, \ldots, 1$, the statistics for each edge $(\hat{s}^{k-1}, \hat{x}^k)$ along the trajectory are updated by:

$$Q\left(\hat{s}^{k-1}, \hat{x}^k\right) = \frac{N\left(\hat{s}^{k-1}, \hat{x}^k\right) \cdot Q\left(\hat{s}^{k-1}, \hat{x}^k\right) + G^k}{N\left(\hat{s}^{k-1}, \hat{x}^k\right) + 1}$$
$$N\left(\hat{s}^{k-1}, \hat{x}^k\right) = N\left(\hat{s}^{k-1}, \hat{x}^k\right) + 1 \quad (38)$$

where, G^k represents the $l-k$-step estimation of the cumulative discounted reward, bootstrapping from the value v^l,

$$G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau \hat{r}^{k+1+\tau} + \gamma^{l-k} v^l \quad (39)$$

In (39), γ is the discount factor and $0 < \gamma < 1$. For the online optimization problem of this article, the value functions $Q(\hat{s}, \hat{x})$ is unbounded, which makes the pUCT rule shown in (34) cannot perform properly. To avoid this, a normalized value $\bar{Q} \in [0, 1]$ is adopted in the pUCT rule. $\bar{Q}$ is computed by using the minimum-maximum values observed in the search tree up to that point:

$$\bar{Q}\left(\hat{s}^{k-1}, \hat{x}^k\right) = \frac{Q\left(\hat{s}^{k-1}, \hat{x}^k\right) - \min_{\hat{s}, \hat{x} \in Tree} Q\left(\hat{s}, \hat{x}\right)}{\max_{\hat{s}, \hat{x} \in Tree} Q\left(\hat{s}, \hat{x}\right) - \min_{\hat{s}, \hat{x} \in Tree} Q\left(\hat{s}, \hat{x}\right)} \quad (40)$$

In general, the proposed RM online optimization algorithm utilizes the MCTS algorithm and a learned model to search over hypothetical future trajectories $\hat{x}^1, \ldots, \hat{x}^k$ given historical state observations $s_1, \ldots, s_t$, and outputs a recommended policy π_t and value estimation v_t. Then, a charge/discharge decision is sampled from the policy π_t, and we use SOCP technique to obtain the remaining decisions. The process is repeated till the end of the optimization horizon. Inside the search tree, the representation network is used to generate an initial internal state in order to improve the prediction network performance. The policy and value estimates computed by prediction network are used by each internal node to select and build its child nodes, and the dynamic network is adopted to compute the next state $\hat{s}$ and reward $\hat{r}$ after taking a decision $\hat{x}$. It can be found that, with the help of the learned model, the agent does not use forecast information of renewable energy generation and load power during the decision making process, and can make decisions without prior knowledge of the uncertainties in the microgrid system.

D. Designation of the Neural Network Model

From **Algorithm 1**, one can find that the learned model is critical for the proposed algorithm. To obtain a good model, two things are very important. The first one is to design a suitable network architecture for the representation network, dynamic network, and prediction network. The second one is the off-line training of the designed model. In this subsection, we will design an appropriate neural network model architecture for the optimization problem in this work, while the model training method will be presented in the next subsection.

The internal state is computed using the representation network, which will affect the prediction accuracy of the future quantities (including policies, values, and rewards). So, the representation network needs to be carefully designed. In [1], the convolution neural network and residual blocks are adopted to play Atari games since the observations have large spatial resolutions and a strong spatial correlation. However, different from Atari games, the observations of the RM system have strong temporal correlation. For instance, we find strong correlations between wind/solar power in adjacent time periods. Considering this, we propose to use Long Short-Term Memory (LSTM) networks to extract features from the historical PV generation power, wind power, and load power. Then the current system state shown in (1) is concatenated with the extracted features and will be fed to a multi-layer neural network. The layout of the representation network is shown in
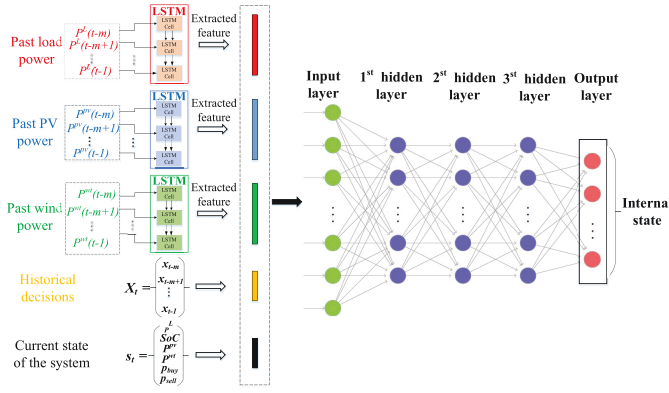
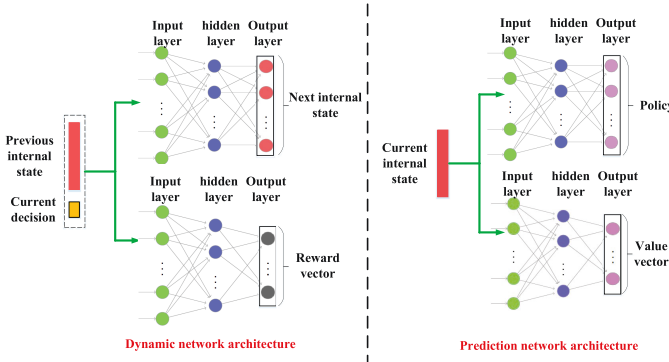Fig. 3.   The architecture of the designed representation network.

Fig. 4.   The architecture of the designed prediction network and the dynamic network.

Fig. 3. The output of the representation network is an O_{rep}-dimensional vector. As we mentioned in Section III-B, the internal state output by the representation network does not have any physical meaning. Unlike using the LSTM networks to directly forecast the PV/wind/load power in [36], there is no future information of PV/wind/load power is used as the label data during the training process. Hence, this procedure conducted by the representation network is not forecasting future power and it can be just regarded as a feature extraction from historical and current system state.

For the dynamic network shown in Fig. 4, the current decision stacked with the internal state of the previous hypothetical step are set as input, and two individual multi-layer neural networks are adopted to respectively compute the next internal state and the reward. The output of the reward computing network is an O_r-dimensional vector. The dynamic network can be regarded as an approximator of the RM system, which enables the proposed model-based learning algorithm to make optimal decisions without knowledge of the system dynamics.

The prediction network shown in Fig. 4 also contains two individual multi-layer neural networks. One is used to compute the probability of each charge/discharge decision being selected, and the other is to get the value v^k in (31). The output of the network can estimate the possible total rewards after current time-step and predict which of the currently available charge/discharge decisions are likely to work best. The output of the policy computing network and value computing network

are an O_p-dimensional vector and an O_v-dimensional vector, respectively. Note that the prediction network (as named in [1]) here is used to compute the policy **p** and value v to help the agent makes optimal decisions, unlike the neural network based renewable power forecasting module predicting the future wind/solar power. Finally, the designed representation network, dynamic network, and prediction network formed the model that utilized in MCTS.

E. Training Method of the Proposed MB-DRL Algorithm

To obtain a good online optimization performance, the model needs to be well-trained off-line first. The model is trained by reinforcement learning from data of self-play. The parameters of the representation network, dynamic network, and prediction network are trained jointly to accurately match the computed policy, value, and reward, for every hypothetical step k, with corresponding target values observed when k actual time-steps have elapsed. More specifically, the training objective is to minimize the following errors by updating network weights:

$$l_t(\theta, \vartheta, \phi) = \sum_{k=0}^{K}\Big[l^r\big(r_{t+k}, \hat{r}_t^k\big) + l^v\big(z_{t+k}, v_t^k\big) + l^p\big(\pi_{t+k}, p_t^k\big)$$
$$+ c\big(\|\theta\|^2 + \|\vartheta\|^2 + \|\phi\|^2\big)\Big] \quad (41)$$

where, r_{t+k} represents the improved reward target, that is, the observed reward after an actual time-step. $\hat{r}_t^k$ is the predicted reward that computed by dynamic network. z_{t+k} represents the improved value target that can be computed by adding up n step discounted rewards and the corresponding search value, $z_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n v_{t+n}$. π_{t+k} represents the improved policy target that is generated by an MCTS search. v_t^k, p_t^k are the predicted value and policy that computed by prediction networks, respectively. So, the first three parts of the right-hand side of the equation (41) represents the errors between the predicted reward/value/policy and the reward/value/policy target. The errors can be computed as (42). The last part of (41) represents the L2 regulation term of the network weights.

$$\begin{cases} l^r(r, \hat{\mathbf{r}}) = \varphi(\varrho(r))^T \log \hat{\mathbf{r}} \\ l^v(z, \mathbf{q}) = \varphi(\varrho(z))^T \log \mathbf{q} \\ l^p(\pi, \mathbf{p}) = \pi^T \log \mathbf{p} \end{cases} \quad (42)$$

where, $\hat{\mathbf{r}}$ is the reward vector computed by the dynamic network; **p** and **q** are the policy and value vector computed by the prediction network. $\varrho(r)$ is an invertible function that used to scale the number r, where $\varrho(r) = sign(r)(\sqrt{|r|+1}-1+\epsilon r)$ with $\epsilon = 10^{-3}$. The function $\varrho(\cdot)$ is introduced to reduce the variance of the optimization target, which can help to improve algorithm convergence [37]. $\varphi(\cdot)$ refers to the transformation function that used to transfer the scalar reward and value targets to equivalent categorical representations. If we use a discrete support set of size $2n+1$ with one support for every integer between -n and n, by applying $\varphi(b)$, a real number b $(-n \le b \le n)$ can be represented through a linear combination of its two adjacent integers $\lfloor b \rfloor$ and $\lceil b \rceil$, so that the original scalar can be recovered by $b = \lfloor b \rfloor \cdot (\lceil b \rceil - b) + \lceil b \rceil \cdot (b - \lfloor b \rfloor)$.
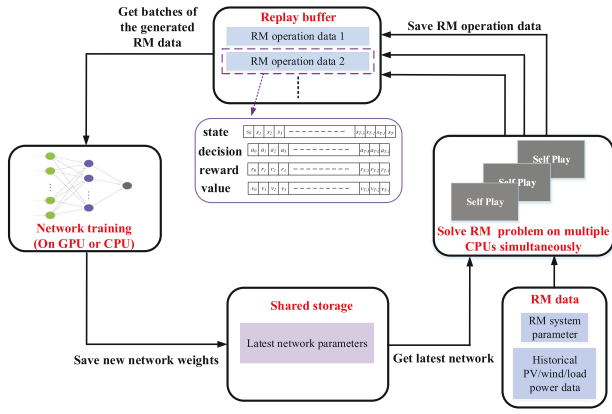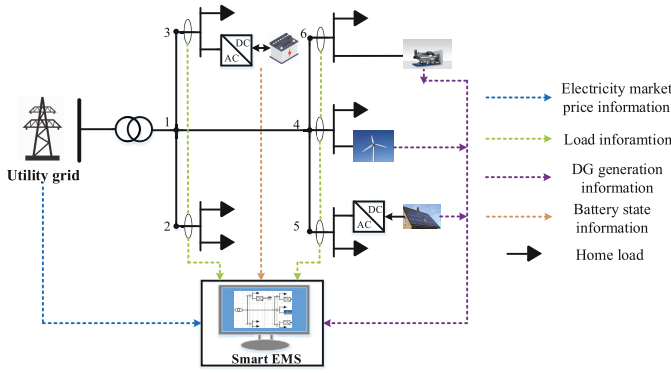
Fig. 5. The training architecture of the model.

TABLE I
THE MARKET ENERGY PRICE AND THE SELLING PRICE ($/kWh)

Time periods	8:00 - 14:00	14:00 - 20:00	20:00 - 22:00	22:00 - 8:00
Market energy price	0.28	0.48	0.28	0.12
Selling price	0.14	0.24	0.14	0.06

TABLE II
THE NETWORK RESISTANCE AND REACTANCE PARAMETERS

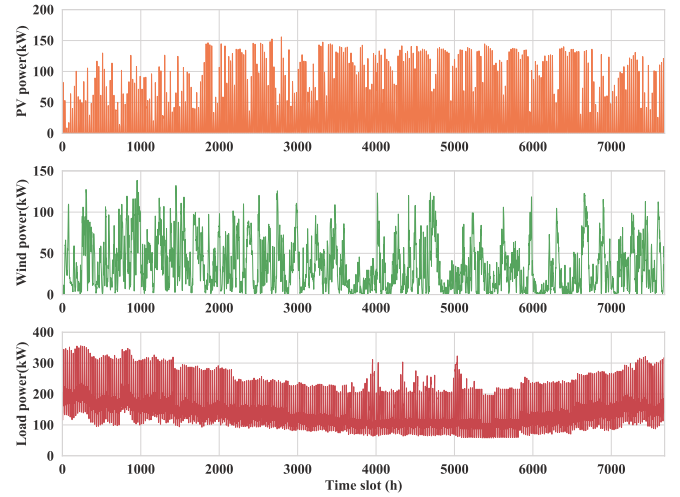| Cable	From bus	To bus	Resistance ($10^{-2}\Omega$)	Reactance ($10^{-2}\Omega$)
L_1	1	2	0.922	0.470
L_2	1	3	4.930	2.511
L_3	1	4	3.660	1.864
L_4	4	5	3.811	1.941
L_5	4	6	1.872	6.188

Fig. 6. The residential microgrid system.

Fig. 7. The training data profiles of solar power, wind power, and load power. The historical wind and PV power data [40] and the residential load data [38] from Jan. 1, 2016 to Nov. 15, 2016 are used as training data set.

Besides, to maintain similar gradient magnitude across different unroll steps, we adopted the gradient scaling method proposed in Appendix G of reference [1].

The model training process is shown in Fig. 5. The training can be split into two independent parts: network training (producing an improved neural network model) and self-play (generating RM operation data). The generated self-play data is stored in the replay buffer, and we sample training data from the buffer to update the neural network model (see Eq. (41)–(42)). Then, the updated neural network is stored in the shared storage, and a random microgrid scenario will be selected by the self-play units to generate new RM operation data with the latest network. The process repeated until the algorithm converged. To speed up the training, network training and self-play can be performed in parallel. The pseudocode of the training method of the proposed algorithm has been given in **Algorithm 2** - **Algorithm4** (see Appendix).

IV. NUMERICAL ANALYSIS

In this section, the online optimization performance of the proposed algorithm is validated on an RM test system shown in Fig. 6.

In the RM system, the electricity is provided by the distributed renewable energy, diesel generator, energy storage system (ESS), and the utility grid. The diesel generator is connected to bus 6, and its upper and lower generation power are 30 kW and 10 kW, respectively. The fuel cost coefficients of the diesel generator are $\alpha_g = 1.04$ $/kW^2h$, $\beta_g = 0.03$ $/kWh$, $c_g = 1.3$ $/h$. The ESS is a 500 kWh @100kW battery with a round-trip efficiency of 90.25%, which is connected to bus 3. To ensure the life span of the battery, the minimum stored energy in the battery is set to be 100 kWh. The degradation cost coefficient is set to be 0.1 $/kWh [25]. The charging/discharging power of the battery is divided into 9 levels $(-100, -75, -50, -25, 0, 25, 50, 75, 100)$ kW. The negative value represents the charging mode. The load data used in the simulation are the historical residential load profiles of Anchorage Alaska State from [38], [39]. To simplify the simulation, we assume that the power demand of each bus accounts for a fixed proportion of the total demand, and the power factor of each load is constant. The active power demand ratio of bus 2 to 6 are 20%, 10%, 30%, 20% and 20%, respectively. Solar power and wind power data are actual historical data from [40]. The profiles of the data are shown in Fig. 7 and Fig. 8. The dynamic market energy price of Southern California residential area [41] is adopted, and the selling price is set to be 50% of the market price, as shown in Table I. The power network resistance and reactance parameters are given in Table II.
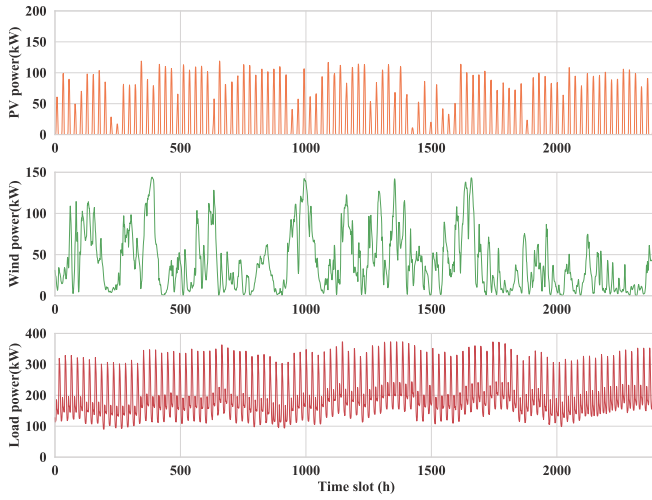
Fig. 8. The testing data profiles of solar power, wind power, and load power. The historical wind and PV power data [40] from Nov. 7, 2015 to Dec. 31, 2015 and Nov. 16, 2016 to Dec. 31, 2016 are used as testing data set. The historical load power data [38] from Nov. 16 to Dec. 31, and the historical load power data [39] from Jan. 1 to Feb. 24 are used as testing data set.

TABLE III
THE NETWORK PARAMETERS OF THE PROPOSED ALGORITHM

Model	Architecture
Representation network	**LSTM**: *input_dim = 6, hidden_nodes = 32* **Fully connected NN**: *input_dim = 32×3 + 6 + 6, Output_dim = 10, hidden layers = [32, 32]*
Dynamic network	**Reward network**: *input_dim = 10 + 1, Output_dim = 10, hidden layers = [32]* **Internal state network**: *input_dim = 10 + 1, Output_dim = 10, hidden layers = [32]*
Prediction network	**Policy network**: *input_dim = 10, Output_dim = 9, hidden layers = [32]* **Value network**: *input_dim = 10, Output_dim = 10, hidden layers = [32]*

In this work, the encoding size of the internal state is set to 10, which means the internal state computed by representation network and dynamic network is a 10-dimensional vector. The support sizes of the reward and value that computed by the prediction network are also set to 10 in this simulation. Three independent LSTM units are adopted in the representation network to extract features from the past 6-h PV power, wind power, and active load power, respectively. The network parameters adopted in the simulation are shown in Table III. The hyperparameters of the algorithm are set as follows: batch size $B = 64$, learning rate $\alpha = 0.005$, discount factor $\gamma = 0.997$, number of simulations per search $N = 20$, number of unrolled hypothetical steps $K = 5$, bootstrapping steps $n = 10$, ratio of self-play speed to training speed is 0.1. All the simulations are conducted on an Intel Core i7-4790 @ 3.60 Ghz × 8 Ubuntu based minitower computer with 32 GB RAM. For the online optimization problem, we use 1 CPU for training and 7 CPUs for self-play. The code is written in Python with PyTorch.

A. Training Result of the Proposed Algorithm

During the model training process, the network updating worker and self-play workers in Fig. 5 are running parallelly
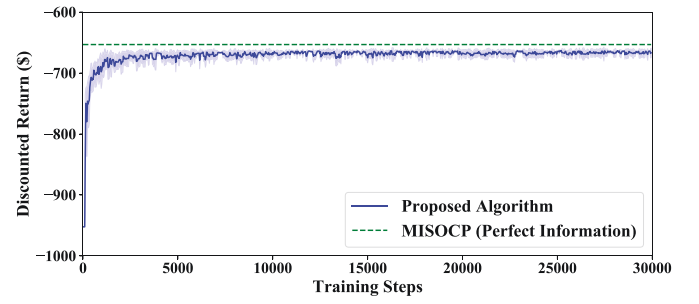

Fig. 9. The convergence process of the proposed algorithm. Blue solid line indicates median returns across 5 separate training runs. The dash line is the result of the MISOCP method under the perfect information. The y-axis represents the average discounted returns for the 10 validation days. The optimal discounted returns optimized by MISOCP is −653.258 dollars.

on 8 CPUs simultaneously with the application of the *Ray* package [42]. The proposed algorithm is trained for 30,000 steps to learn the optimal operational strategy. In the training process, the performance of the algorithm is evaluated every 20 training steps on the selected 10-day validation data set. Fig. 9 illustrates the convergence process of the proposed algorithm across 5 separate runs. On average, the training process of a single training run takes about 1 hour and 35 minutes on the computer mentioned above. It can be seen that the discounted returns increase rapidly with the training step increasing from 0 to 2000. After that the discounted returns increase slowly and finally converge around −665 dollars with small oscillations.

For each validation day, the optimal discounted returns can be obtained by using MISOCP optimization method with the assumption that the perfect RM state information is available. That means the PV and wind power generation, and load power of each time-step is known in prior, which is not realistic in the actual online optimization process. The obtained optimal returns are shown in Fig. 9 as the reference to evaluate the training performance of the proposed algorithm. As shown in Fig. 9, the discounted returns of the proposed algorithm approach rather than reach the optimal value. This is because the proposed algorithm makes the online operation decisions only according to the current system state, without accurate information of the future renewable generations and electricity demand.

B. The Effect of Number of Simulations Per Search on the Algorithm

Training with a different N (number of simulations per search) will result in a different neural network model. In this section, the influence of N on the performance of the proposed algorithm is analyzed when N is set to 5, 10, and 20, respectively. Also, the convergence performance is evaluated across 5 separate training runs for each N setting, and the results are shown in Fig. 10. It can be found that a larger number of simulations per search will improve the performance of the algorithm. However, the improvement becomes less obvious when N increases to a certain threshold.
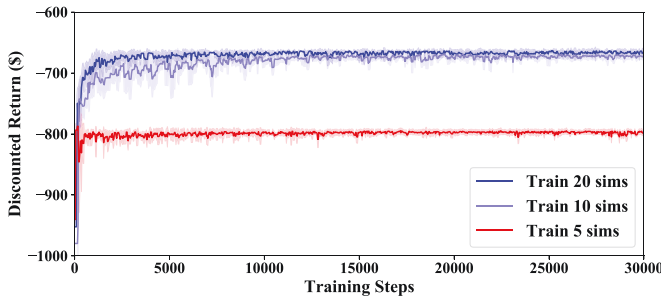
Fig. 10. The convergence process of the proposed algorithm at different numbers of simulations per move in tree search. Solid line indicates median returns across 5 separate training runs.
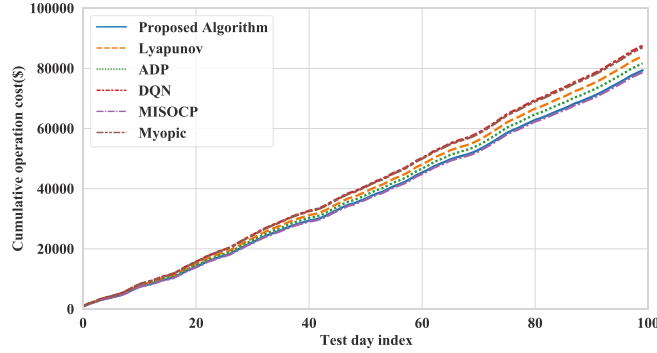

Fig. 11. Cumulative operation cost of the system on the testing data set.

C. Online Optimization Performance of the Proposed Algorithm

After the off-line training of the neural network model is completed, the proposed algorithm can be applied online to schedule the microgrid, as shown in **Algorithm 1**. For the online scheduling application, the trained model in Section IV-A is loaded and the number of simulations per search N is set to 20. To validate the effectiveness of the proposed algorithm, the comparison is conducted between the proposed algorithm and other state-of-the-art online optimization approaches, such as deep Q network (DQN), Lyapunov optimization [19], [43], ADP [24], myopic policy. In this work, the myopic policy optimizes operation cost now, but do not explicitly consider the influence of the current decisions on the cumulative operation cost of the RM in the future. 100-day data shown in Fig. 8 is used to test the performance, and the cumulative operational cost of the system optimized by different algorithms are calculated, as shown in Fig. 11. The average daily operation cost of the RM system optimized by the proposed method, Lyapunov optimization, ADP, DQN, and myopic policy are $ 804.08, $ 839.65, $ 815.79, $ 874.56, and $ 869.34, respectively. Besides, the average daily operation cost optimized using the MISOCP is $ 786.35. The results indicate that the proposed algorithm outperforms myopic, Lyapunov optimization, ADP, and DQN algorithms in terms of the daily operation cost on average by 7.52%, 4.24%, 1.44%, and 8.06%, respectively, and yields an operation cost that closely follows the optimal cost (is only 2.25% higher) computed by the MISOCP method under perfect information.
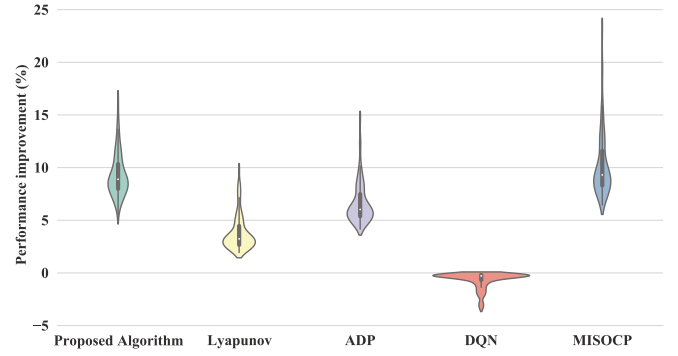

Fig. 12. Violin plot of the performance improvement of different optimization methods compared to myopic method.

TABLE IV
THE PERFORMANCE IMPROVEMENT OF DIFFERENT OPTIMIZATION
METHODS COMPARED TO MYOPIC POLICY ON THE TESTING DATA SET

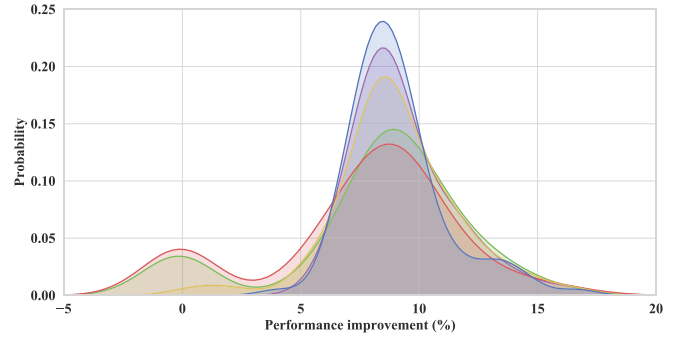| *Performance improvement*		*Mean*	*Maximum*	*Minimum*	*Standard deviation*
Online methods	Proposed algorithm	**9.30%**	16.68%	5.28%	2.12%
	Lyapunov optimization	3.76%	9.89%	1.93%	1.65%
	ADP	6.57%	14.78%	4.16%	1.92%
	DQN	-0.65%	-0.14%	-3.45%	0.77%
Off-line method	MISOCP	10.20%	23.28%	6.45%	3.02%

Fig. 13. The distribution of the performance improvement of the proposed algorithm compared to myopic policy in 5 separate runs.

Using the result optimized by myopic policy as the baseline, the performance improvement of different methods are further evaluated, as shown in Fig. 12 and Table IV. Fig. 12 shows the comparison of the performance improvement of the adopted algorithms. In Table IV, the statistical indicators of the performance improvement of different online optimization methods are presented. It can be found that the proposed algorithm obtains the greatest performance improvement among the online optimization algorithms. The DQN algorithm performs worse than the Lyapunov optimization and the ADP algorithm. We attribute the poor performance of the DQN algorithm to the low sample efficiency of the policy, and the non-stationary environment. Although MISOCP method performs best, it is an off-line optimization algorithm and the optimal scheduling under perfect information can never be achieved since we cannot accurately forecast the future state of the microgrid system.

TABLE V
THE PERFORMANCE IMPROVEMENT OF THE PROPOSED ALGORITHM
COMPARED TO MYOPIC POLICY IN 5 INDIVIDUAL RUNS

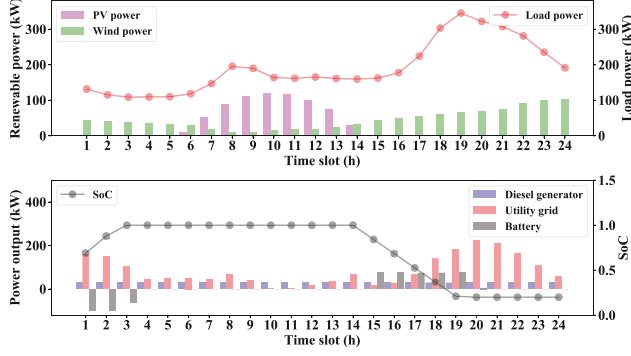| Performance improvement	Mean	Maximum	Minimum	Standard deviation
#1	8.22%	16.68%	-0.17%	3.83%
#2	9.30%	16.68%	5.28%	2.12%
#3	8.60%	16.67%	-0.17%	3.99%
#4	9.04%	16.68%	0.33%	2.59%
#5	9.05%	16.68%	4.05%	2.04%

Fig. 14. The online energy scheduling results of the proposed algorithm.

D. Repeatability and Feasibility of the Proposed Algorithm

To validate the repeatability of the proposed algorithm, 5 individual off-line training runs are conducted and 5 corresponding neural network models are obtained. Then the online optimization performance is evaluated by applying each model on the testing data set individually. The simulation results of the 5 individual runs are shown in Table V. It can be found that the proposed algorithm performs very stable.

To validate the feasibility of the decisions made by the proposed approach, the online scheduling details including the SoC pattern, power exchange between the RM and utility grid, charge/discharge power of the battery, and the power output of diesel generator are shown in Fig. 14. It can be observed that the proposed algorithm has learned to charge the battery when the electricity price is low and to discharge when the price is on-peak, and also learned to dispatch controllable generators. Besides, the SoC patterns of the other online optimization algorithms and the optimal pattern are illustrated in Fig. 15. From the results in Fig. 14 and Fig. 15, it can be found that the proposed algorithm and the ADP method almost learned the optimal SoC pattern, while the other online optimization algorithms performs worse than the two algorithms. Also, the relaxation gap is calculated by $| (P_{ij}^2 + Q_{ij}^2)/v_i - l_{ij} |$ and plotted in Fig. 16. Note that smaller values of the gap mean better AC power flow feasibility [44]. The maximum gap in Fig. 16 is less than 10^{-6}, which validates the effectiveness of the scheduling results in Fig. 14. Finally, according to the simulation results, the proposed online optimization algorithm takes an average of 0.07s to make one single time-step scheduling, which can fulfill the time requirements of the real-time application. The average time consumption of the Lyapunov optimization, ADP, DQN, and myopic policy to make one single time-step scheduling are 1.18s, 1.71s, 0.0038s, and 1.56s,
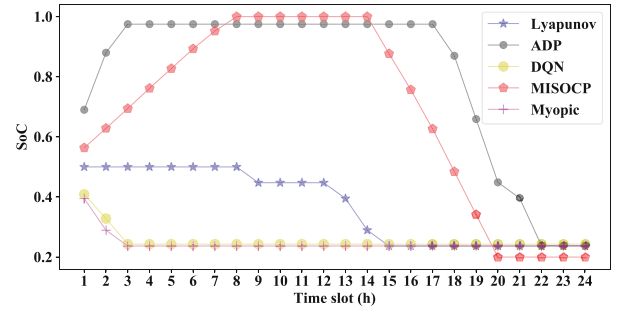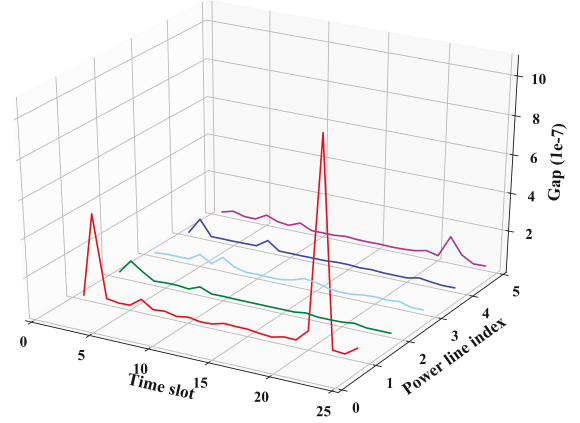

Fig. 15. The SoC optimized by the other algorithms.

Fig. 16. The relaxation gaps of the power flow.

Algorithm 2 The Training Process of the RM Online Optimization Algorithm

---

1: Initialize the neural networks of the model; set the total number of training steps N_T, number of simulations per search N, discount factor γ, batch size B, replay buffer size W, unroll steps K, bootstrapping steps n, ratio of self-play speed to training speed υ_s, and other hyperparameters. Initialize the training step index $n_s = 0$.

2: Initialize self-play workers, training worker, replay buffer worker, and shared storage workers, then launch workers. ▷ *Self-play workers run on several CPUs in parallel.*

3: **while** $n_s \leq N_T$ **do**:
    1) $D \leftarrow$ *SELF-PLAY()*. ▷ *Generate RM operation data by self-play workers.*
    2) *REPLAY BUFFER.ADD(D)*. ▷ *Save the generated operation data in the replay buffer.*
    3) $(\theta, \vartheta, \phi) \leftarrow$ *SHARED STORAGE.LATEST()*. ▷ *Get latest model parameters.*
    4) $B \leftarrow$ *REPLAY BUFFER.SAMPLE()*. ▷ *Sample batch data B from the replay buffer worker.*
    5) $(\theta_{new}, \vartheta_{new}, \phi_{new}) \leftarrow$ *NETWORK TRAINING($\theta, \vartheta, \phi, B, n_s$)* ▷ *Update the latest model.*
    6) *SHARED STORAGE.ADD($\theta_{new}, \vartheta_{new}, \phi_{new}$)* ▷ *Save the updated model parameters in the shared storage worker.*
    7) $n_s = n_s + 1$

4: **end while**

5: Return the latest neural network model.

---

respectively. It can be found that the online optimization efficiency of the proposed algorithm is higher than most of the above algorithms.

Algorithm 3 SELF-PLAY

Input: The model; the PV power, wind power, and load power scenario; the training index n_S;

Output: The generated RM operation data.   ▷ *The procedure includes 3 parts: get the latest model from the shared storage, play game, save game to replay buffer.*

1: Load $(\theta, \vartheta, \phi)$. ▷ *Get the latest model from the shared storage*
2: Randomly select a day from the training data set and load the data. ▷ *PV/wind sequence, load sequence, and price sequence.*
3: Reset the RM simulation environment, and get the initial state s_0.
4: **for** $t = 0, 1, 2, \ldots, T - 1$ **do**:   ▷ *T = 24 in this work.*
5:      Internal state $\hat{s}_t \leftarrow h_\theta(s_t, s_{t-1}, \ldots, s_0)$ ▷ (29).
6:      Create root node γ_{root} with state $\hat{s}_t$.
7:      Run **MCTS**$(\gamma_{root}, \theta, \vartheta, \phi, \chi)$.   ▷ *χ is the decision history.*
8:      Select $P_b(t)$ that leads to the most visited child of root node.
9:      Overcharge/overdischarge check.   ▷ *(33).*
10:      Solve the OPF sub-problem to get the remaining decisions.
11:      Apply the optimal decision to the RM, and get actual reward r_t and the next state s_{t+1}.   ▷ (18), (28).
12:      Store search statistics.   ▷ *visit counts, root value, etc.*
13: **end for**
14: If the ratio of replay buffer data size to the shared storage data size is greater than υ_s, pause the self-play worker for 0.5 seconds.
15: Return generated RM operation data.   ▷ $(s_0, x_0, r_0, v_0, s_1)$, $(s_1, x_1, r_1, v_1, s_2), \cdots$
16:
17: **function MCTS**$(\gamma_{root}, \theta, \vartheta, \phi, \chi)$

18:      **for** $n_{sim} \in 1, 2, \ldots, N$ **do**
19:          $node \leftarrow \gamma_{root}$.
20:          history $= \chi$, $SearchPath = [node]$.
21:          **while** $node$ is not leaf node **do**
22:              Select the child node with highest UCB score, and get the charge/discharge decision $\hat{x}_k$.   ▷ (34).
23:              history.append($\hat{x}^k$)
24:              $node \leftarrow$ new child node.
25:              $SearchPath$.append($node$).
26:          **end while**
27:          $parent \leftarrow SearchPath[-2]$.
28:          Using the dynamic network, compute the reward $\hat{r}^k$ and the next internal state $\hat{s}^{k+1}$ after taking decision history[-1] from $parent$.   ▷ (35)
29:          Using the prediction network, compute the policy p^{k+1} and value v^{k+1}.   ▷ (36)
30:          According to $\hat{s}^{k+1}$, $\hat{r}^k$, p^{k+1}, and v^{k+1}, update parameters of the $node$ and expand $node$.
31:          **for** $node$ in reversed($SearchPath$) **do**
32:              value $= v^{k+1}$
33:              $node.value = node.value +$ value.
34:              $node.visit\_count = node.visit\_count + 1$.
35:              value $= node.reward + \gamma *$ value.   ▷ (38) -(40)
36:          **end for**
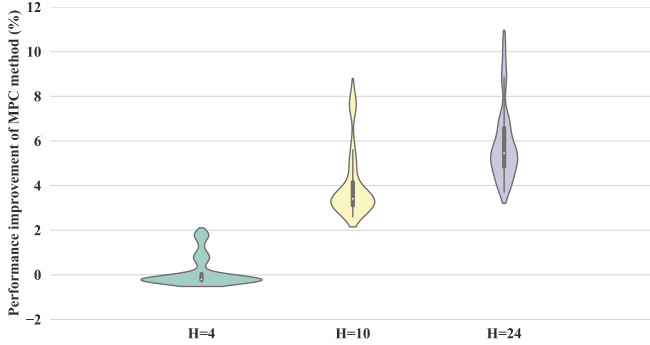37:   **end for**
38: **end function**

Fig. 17. Violin plot of the performance improvement of the MPC method compared to myopic policy.

E. Comparison of the Proposed Algorithm and Model Predictive Control (MPC) Method

In this section, the simulations of the online scheduling using classical MPC method are conducted and compared to the proposed algorithm. MPC makes online decisions according to the near future prediction information from a forecast model. For the MPC method, the prediction errors of the PV/wind power and load are set to 10%, and 3%, respectively. The looking forward window (H) of the MPC method are set to $4h$, $10h$, and $24h$, respectively. Fig. 17 shows the online optimization performance improvement of the MPC method under different H values. As shown in Fig. 17, the average online optimization performance improvements of the MPC method compared to myopic policy are 0.13%, 4.03%, and 5.84%, respectively. It can be found that, the performance improvement of the MPC method increases with the increase

of H. However, it still underperforms the proposed algorithm (9.30% as shown in Table IV), even if the forecasting information of PV/wind/load power in the next 24 hours are utilized. Besides, the average time consumption of the MPC method ($H = 24h$) to make one single time-step scheduling is 3.83s, which is much longer than the time needed by the proposed algorithm.

V. CONCLUSION

This article investigates the optimization algorithm for the online optimization of an RM under uncertainties, which is based on a newly developed MB-DRL algorithm, MuZero [1]. It combines a Monte-Carlo tree search method with a learned neural network model which consists of representation network, dynamic network, and prediction network. To deal with the huge decision space and constraints in the RM model, a framework was proposed to combine the MB-DRL method with the SOCP technique. With the consideration of the characteristics of the optimization problem in this work, the representation network is redesigned to adopt LSTM units to extract features from historical RM state data and feed into a fully connected neural network. The parameters of these networks are updated by gradient descent using the operation data generated by the self-play mechanism during the training process. Numerical simulation results show that the proposed approach can learn to operate the microgrid by itself from data of self-play, and can make online decisions without dependence on renewable and load power prediction information from forecast models. Besides, the proposed algorithm outperforms the state-of-the-art online optimization methods,

Algorithm 4 Network Training

Input: The model parameter θ, ϑ, ϕ, B; the training index n_s.
Output: The updated model.
  1: Set the learning rate and the optimizer.   ▷ *Constant learning rate in this article.*
  2: Load (θ, ϑ, ϕ).   ▷ *Get the latest model from the shared storage*
  3: Get the training data B.
  4: According to the batch data B, compute the evaluation of the value, reward, policy, and internal state using the model (θ, ϑ, ϕ).
  5: Get the target from the batch B, then calculate the cross entropy loss using (41) and (42).
  6: Update the model parameters using the Adam optimizer.
  7: If the ratio of replay buffer data size to the shared storage data size is less than υ_s, pause the network training worker for 0.5 seconds.
  8: Return the updated model.

---

including the DQN, Lyapunov optimization, ADP, myopic policy, and MPC method.

APPENDIX

See the Algorithms 2–4.

REFERENCES

[1] J. Schrittwieser *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," 2019. [Online]. Available: arXiv:1911.08265.

[2] D. T. Ton and M. A. Smith, "The U.S. department of energy's microgrid initiative," *Elect. J.*, vol. 25, no. 8, pp. 84–94, 2012.

[3] S. Parhizi, H. Lotfi, A. Khodaei, and S. Bahramirad, "State of the art in research on microgrids: A review," *IEEE Access*, vol. 3, pp. 890–925, 2015.

[4] J. I. Giraldez Miner, F. Flores-Espino, S. MacAlpine, and P. Asmus, "Phase I microgrid cost study: Data collection and analysis of microgrid costs in the United States," Nat. Renew. Energy Lab. (NREL), Golden, CO, USA, Rep. NREL/TP-5D00-67821, 2018. [Online]. Available: https://www.nrel.gov/docs/fy19osti/67821.pdf

[5] "Projects and trends in the global microgrid market by region, segment, business model, and top states and countries," Navigant Res., Boulder, CO, USA, Microgrid Deployment Tracker Rep. 1Q20, 2020. [Online]. Available: https://guidehouseinsights.com/reports/microgrid-deployment-tracker-1q20

[6] M. F. Zia, E. Elbouchikhi, and M. Benbouzid, "Microgrids energy management systems: A critical review on methods, solutions, and prospects," *Appl. Energy*, vol. 222, pp. 1033–1055, Jul. 2018.

[7] H. Lei, S. Huang, Y. Liu, and T. Zhang, "Robust optimization for microgrid defense resource planning and allocation against multi-period attacks," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5841–5850, Sep. 2019.

[8] M. Yan, M. Shahidehpour, A. Paaso, L. X. Zhang, A. Alabdulwahab, and A. Abusorrah, "Distribution network-constrained optimization of peer-to-peer transactive energy trading among multi-microgrids," *IEEE Trans. Smart Grid*, early access, Oct. 22, 2020, doi: 10.1109/TSG.2020.3032889.

[9] H. Alharbi and K. Bhattacharya, "Stochastic optimal planning of battery energy storage systems for isolated microgrids," *IEEE Trans. Sustain. Energy*, vol. 9, no. 1, pp. 211–227, Jan. 2018.

[10] C. Yuan, M. S. Illindala, and A. S. Khalsa, "Co-optimization scheme for distributed energy resource planning in community microgrids," *IEEE Trans. Sustain. Energy*, vol. 8, no. 4, pp. 1351–1360, Oct. 2017.

[11] G. Liu, Y. Xu, and K. Tomsovic, "Bidding strategy for microgrid in day-ahead market based on hybrid stochastic/robust optimization," *IEEE Trans. Smart Grid*, vol. 7, no. 1, pp. 227–237, Jan. 2016.

[12] H. Farzin, M. Fotuhi-Firuzabad, and M. Moeini-Aghtaie, "A stochastic multi-objective framework for optimal scheduling of energy storage systems in microgrids," *IEEE Trans. Smart Grid*, vol. 8, no. 1, pp. 117–127, Jan. 2017.

[13] T. Wang, H. Kamath, and S. Willard, "Control and optimization of grid-tied photovoltaic storage systems using model predictive control," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 1010–1017, Mar. 2014.

[14] X. Yang, Y. Zhang, H. He, S. Ren, and G. Weng, "Real-time demand side management for a microgrid considering uncertainties," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3401–3414, May 2019.

[15] H. Shuai, J. Fang, X. Ai, W. Yao, J. Wen, and H. He, "On-line energy management of microgrid via parametric cost function approximation," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 3300–3302, Jul. 2019.

[16] L. Lu, J. Tu, C.-K. Chau, M. Chen, and X. Lin, "Online energy generation scheduling for microgrids with intermittent energy sources and co-generation," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 53–66, 2013.

[17] Y. Jia, X. Lyu, C. S. Lai, Z. Xu, and M. Chen, "A retroactive approach to microgrid real-time scheduling in quest of perfect dispatch solution," *J. Mod. Power Syst. Clean Energy*, vol. 7, no. 6, pp. 1608–1618, 2019.

[18] Y. Jia, X. Lyu, P. Xie, Z. Xu, and M. Chen, "A novel retrospect-inspired regime for microgrid real-time energy scheduling with heterogeneous sources," *IEEE Trans. Smart Grid*, vol. 11, no. 6, pp. 4614–4625, Nov. 2020.

[19] W. Shi, N. Li, C.-C. Chu, and R. Gadh, "Real-time energy management in microgrids," *IEEE Trans. Smart Grid*, vol. 8, no. 1, pp. 228–238, Jan. 2017.

[20] W.-J. Ma, J. Wang, V. Gupta, and C. Chen, "Distributed energy management for networked microgrids using online ADMM with regret," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 847–856, Mar. 2018.

[21] B. Li, T. Chen, X. Wang, and G. B. Giannakis, "Real-time energy management in microgrids with reduced battery capacity requirements," *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 1928–1938, Mar. 2019.

[22] K. Rahbar, J. Xu, and R. Zhang, "Real-time energy storage management for renewable integration in microgrid: An off-line optimization approach," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 124–134, Jan. 2015.

[23] H. Shuai, J. Fang, X. Ai, Y. Tang, J. Wen, and H. He, "Stochastic optimization of economic dispatch for microgrid based on approximate dynamic programming," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 2440–2452, May 2019.

[24] H. Shuai, J. Fang, X. Ai, J. Wen, and H. He, "Optimal real-time operation strategy for microgrid: An ADP-based stochastic nonlinear optimization approach," *IEEE Trans. Sustain. Energy*, vol. 10, no. 2, pp. 931–942, Apr. 2019.

[25] Y. Du and F. Li, "Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1066–1076, Mar. 2020.

[26] J. Gao, "Machine learning applications for data center optimization," Google, Mountain View, CA, USA, White Paper, 2014.

[27] L. Lei, Y. Tan, G. Dahlenburg, W. Xiang, and K. Zheng, "Dynamic energy dispatch in isolated microgrids based on deep reinforcement learning," 2020. [Online]. Available: https://arxiv.org/abs/2002.02581.

[28] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *Proc. Eur. Workshop Reinforcement Learn. (EWRL)*, pp. 1–7, 2016.

[29] Y. Ji, J. Wang, J. Xu, X. Fang, and H. Zhang, "Real-time energy management of a microgrid using deep reinforcement learning," *Energies*, vol. 12, no. 12, p. 2291, 2019.

[30] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[31] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[32] W. Liu, P. Zhuang, H. Liang, J. Peng, and Z. Huang, "Distributed economic dispatch in microgrids based on cooperative reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2192–2203, Jun. 2018.

[33] S. H. Low, "Convex relaxation of optimal power flow—Part I: Formulations and equivalence," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 15–27, Mar. 2014.

[34] C. B. Brown *et al.*, "A survey of Monte–Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[35] J. Rajamäki and P. Hämäläinen, "Continuous control Monte–Carlo tree search informed by multiple experts," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 8, pp. 2540–2553, Aug. 2019.

[36] Y. Su, J. Yu, M. Tan, Z. Wu, Z. Xiao, and J. Hu, "A LSTM based wind power forecasting method considering wind frequency components and the wind turbine states," in *Proc. 22nd Int. Conf. Elect. Mach. Syst. (ICEMS)*, Harbin, China, 2019, pp. 1–6.

[37] T. Pohlen *et al.*, "Observe and look further: Achieving consistent performance on atari," 2018. [Online]. Available: arXiv:1805.11593.

[38] *Training Load Data, USA_AK_Anchorage.Intl.Ap.702730_Tmy3_ High.CSV*. Accessed: Feb. 10, 2020. [Online]. Available: https:// openei.org/datasets/files/961/pub/RESIDENTIAL_LOAD_DATA_E_ PLUS_OUTPUT/

[39] *Testing Load Data USA_AK_Fairbanks.AP.702610_TMY3_ High.CSV*. Accessed: Feb. 10, 2020. [Online]. Available: https://openei. org/datasets/files/961/pub/RESIDENTIAL_LOAD_DATA_E_PLUS_ OUTPUT/.

[40] *Wind and PV Power From Open Power System Data*. Accessed: Feb. 10, 2020. [Online]. Available: https://data.open-power-system-data.org/ ninja_pv_wind_profiles

[41] *Price Data*. Accessed: Feb. 10, 2020. [Online]. Available: https://www. sce.com/wps/portal/home/residential/rates/Time-Of-Use-Residential- Rate

[42] *Ray*. Accessed: Feb. 10, 2020. [Online]. Available: https://ray. readthedocs.io/en/latest/

[43] W. Sheng, P. Li, Q. Duan, Z. Li, and C. Zhu, "Research on energy man- agement strategy of energy hub with energy router based on Lyapunov optimization method," *CSEE J. Power Energy Syst.*, vol. 39, no. 21, pp. 6212–6224, 2019, doi: 10.13334/j.0258-8013.pcsee.181086.

[44] Z. Yuan and M. R. Hesamzadeh, "Second-order cone AC optimal power flow: Convex relaxations and feasible solutions," *J. Mod. Power Syst. Clean Energy*, vol. 7, no. 2, pp. 268–280, 2019.

Haibo He (Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in electrical engineering from Ohio University, Athens, OH, USA, in 2006. From 2006 to 2009, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology. He is currently the Robert Haas Endowed Chair Professor with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. He was a recip- ient of the IEEE International Conference on Communications Best Paper Award in 2014, the IEEE Computational Intelligence Society Outstanding Early Career Award in 2014, and the National Science Foundation Career Award in 2011. He severed as the General Chair of the IEEE Symposium Series on Computational Intelligence in 2014. He is currently the Editor- in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Hang Shuai (Member, IEEE) received the B.Eng. degree from the Wuhan Institute of Technology, Wuhan, China, in 2013, and the Ph.D. degree in elec- trical engineering from the Huazhong University of Science and Technology, Wuhan, in 2019.

He was also a Visiting Student Researcher with the University of Rhode Island, Kingston, RI, USA, from 2018 to 2019. He is currently a Postdoctoral Researcher with the University of Rhode Island. His research interests include deep reinforcement learn- ing, power systems operation and economics, and integrated energy system.