Stochastic Dispatch of Energy Storage in Microgrids: A Reinforcement Learning Approach Incorporated with MCTS

Yuwei Shang, Wenchuan Wu, Senior Member, IEEE, Jianbo Guo, Zhe Lv, Zhao Ma, Wanxing Sheng, Ran Chen

Abstract—The dynamic dispatch (DD) of battery energy storage systems (BESSs) in microgrids integrated with volatile energy resources is essentially a multiperiod stochastic optimization problem (MSOP). Because the life span of a BESS is significantly affected by its charging and discharging behaviors, its lifecycle degradation costs should be incorporated into the DD model of BESSs, which makes it non-convex. In general, this MSOP is intractable. To solve this problem, we propose a reinforcement learning (RL) solution augmented with Monte-Carlo tree search (MCTS) and domain knowledge expressed as dispatching rules. In this solution, the Q-learning with function approximation is employed as the basic learning architecture that allows multistep bootstrapping and continuous policy learning. To improve the computation efficiency of randomized multistep simulations, we employed the MCTS to estimate the expected maximum action values. Moreover, we embedded a few dispatching rules in RL as probabilistic logics to reduce infeasible action explorations, which can improve the quality of the data-driven solution. Numerical test results show the proposed algorithm outperforms other baseline RL algorithms in all cases tested.

Index Terms—microgrid, energy storage, volatile energy resource, dynamic dispatch, reinforcement learning.

NOMENCLATURE

For the management of battery energy storage systems:

- $SOC$: Battery state of charge
- $\sigma$: battery self-discharge rate
- $\eta_B$: battery charging/discharging efficiency
- $P_B$: charging/discharging power of the battery
- $C_B$: capacity of the battery
- $V_B$-nom: rated voltage of the battery
- $Q_{life}$: lifetime throughput of the battery
- $N_f$: number of cycles until failure of the battery
- $P_{min}^b$, $P_{max}^b$: minimum, maximum power of the battery
- $SOC_{min}$, $SOC_{max}$: minimum, maximum state of charge
- $c$: electricity tariff
- $P_{min}^{PCC}$, $P_{max}^{PCC}$: minimum, maximum power at PCC
- $DOD$: Depth of discharge

In the Markov decision process:

- $a$, $s$, $R$: action, state, immediate reward
- $A$, $S$, $\mathcal{R}$: set of all actions, states, rewards
- $\pi$: probability transition function
- $\gamma$: policy (action selection rule)
- $Q(s,a)$: value function for taking action $a$ in state $s$
- $\alpha$: step-size parameter
- $\gamma$: discount-rate parameter
- $G_{t}^{\pi}$: cumulative rewards from $t$ to $t+n$
- $\phi_{X}^{\pi}$: potential function for the domain knowledge
- $A_{t}$: set of feasible actions

I. INTRODUCTION

Volatile energy resources, such as loads from renewable energy based distributed generators (DGs) and electric vehicles (EVs), significantly affect the operation of power systems. In microgrids, we can coordinate volatile energy resources and energy storage to mitigate power fluctuations. Hence, battery energy storage systems (BESSs) are widely used to balance the power and shave peaks in microgrids. Furthermore, BESSs can be scheduled to increase the electricity revenue for microgrid entities by charging energy in low-price periods and discharging energy in high-price periods. Therefore, how to dynamically dispatch the BESS such that the operational costs of the microgrids are minimized while satisfying the operational constraints of the distribution network is a key challenge.

Many studies have focused on the dynamic dispatch of BESSs. Some works employ deterministic optimization models [1–4]. However, due to the stochastic nature of DGs and EVs, the dynamic dispatch of BESSs is essentially a multiperiod stochastic optimization problem (MSOP). One way to solve MSOPs is to apply scenario-based stochastic programming (SP) [5–9]. In this approach, Monte Carlo simulations are employed to repeatedly generate scenarios across a multistep process. The computational burden increases exponentially with the number of scenarios investigated. Additionally, because the life span of a BESS is significantly influenced by its charging and discharging behavior, the lifecycle degradation costs of BESSs should be incorporated into the microgrid optimization.
objectives. However, most of the existing SP models either assume zero degradation costs for the BESS [5–6, 9], or simplify the battery cycle life to a linear function of the Depth of Discharge (DOD) [7, 8], which may introduce additional estimation error on the BESS degradation cost [10]. When a more accurate degradation cost model is used for BESSs, the MSOP generally becomes nonconvex and computationally challenging [11].

Reinforcement learning (RL) may be a viable alternative for tackling an MSOP with a non-convex objective function [11, 12]. RL arose from dynamical systems theory, and is formalized by the Bellmanian Equation and Markov decision process (MDP). A fundamental issue in RL is the balance of exploration and exploitation, which facilitates action-value estimation and policy improvement. It is common for an RL agent to occasionally explore some random actions and learn from experience. In Q-learning, this trial-and-error learning process is guaranteed with asymptotic convergence. As the bootstrapping stage increases, the error of action value estimation decreases (i.e., the error reduction property); yet the conventional learning algorithm suffers from increased computation complexity [13]. To reduce the computation burden of MSOPs, the Monte-Carlo tree search (MCTS) method may be a viable solution which shows remarkable success recently [13]. Motivated by these achievements, we study how to incorporate MCTS into Q-learning to solve the stochastic dispatch of BESS in microgrids.

B. Related work

The early related researchers mainly employ deterministic models for scheduling BESSs. Reference [1] introduces linear programming (LP) to mitigate fluctuations in photovoltaic (PV) output and increase the electricity revenue in the microgrid. In addition to increasing the electricity revenue, the efficiency of the BESS is considered in [2], in which a non-linear optimization model is formulated and solved by a meta-heuristic algorithm. Reference [3] formulates a quadratic programming (QP) to achieve economic microgrid dispatch. A different objective is considered in [4], namely to satisfy the constraints of the distribution network by tracking the power profile established on a day-ahead basis. They formulate a QP and employ model predictive control (MPC) to schedule the BESS. These deterministic models neglect the intermittency and variability of volatile energy resources.

Some other researchers formulate the BESS scheduling problem as stochastic optimization models, which tackles the uncertainties associated with volatile energy resources. In [5], a two-stage stochastic mixed-integer programming (SMIP) is formulated to optimize the dispatching policy for microgrids. In [6], the problem of storage co-optimization is addressed by formulating a two-stage SMIP using piecewise-linear approximation of the value function. In [7], a two-stage stochastic mixed-integer nonlinear model is formulated, and the battery degradation cost was considered by simplifying its cycle life as a linear function of the DOD. In [8], the BESS degradation cost model is formulated as an equivalent fuel-run generator, which enables it to be incorporated into a unit commitment problem. Their battery cycle life model is similar to [7]. In addition to the two-stage SP models, a multistage SP model is formulated in [9], and solved by a customized stochastic dual dynamic programming algorithm.

Besides the above methods, some works have explored RL methods for scheduling BESSs. A cooperative RL algorithm is proposed in [11], whose dispatch objectives incorporate a non-convex BESS degradation cost model. In [12], a dual Q-iterative learning algorithm is proposed to minimize the microgrid operation cost. These methods ignore the uncertainties between state transitions along the multistep bootstrapping trajectories [9].

In the field of machine learning, combining the MCTS method and embedding domain knowledge into data-driven solutions can enhance their performances [14–21], which inspire us on tackling MSOPs. For the MCTS algorithms, [14] introduces its basic idea, in which tree search policies are used to asymptotically focus the Monte Carlo trials on multistep Bootstrapping trajectories that are promising to high-return rewards. [15] presents a survey of different variants of MCTS. [16] adopts the MCTS to tackle the computationally intensive game GO. In the studies of incorporating domain knowledge, [17, 18] demonstrate the performance enhancement of RL solutions by leveraging different kinds of domain knowledge. To numerically express the rule based domain knowledge, the probabilistic soft logic (PSL) is formalized in [19] and employed in [20, 21], where knowledge rules are used to supervise the learning process.

Contributions

We formulate a multiperiod stochastic model for dispatching the BESS in microgrids. The degradation cost model of BESSs adopted in this work is a benchmark model employed in the microgrid simulation tool HOMER [22] and other applications [23, 24]. In our RL based solution, the key for identifying statistically optimal dispatching policies is the estimation of expected maximum action values. This may be achieved by naively computing the optimal value function in the scenario based search trees containing $b^d$ possible sequences of actions, where $b$ is the number of discretized actions per state (tree’s breadth) and $d$ is the number of steps (tree’s depth). So, its computation complexity increases rapidly as the number of scenarios increases. To reduce the computation burden, we employ the MCTS algorithm that has made prolific achievements in playing Go. However, from the perspective of game theory, the MCTS in Go tackles two-player zero-sum deterministic game [15], yet in our case it tackles single-player stochastic power dispatching. How to integrate the MCTS into the Q-learning for solving the MSOP is a key challenge in this work. Moreover, in order to incorporate the domain knowledge for performance enhancement, a knowledge incorporation scheme is needed to numerically express different knowledge rules and combine them for reducing infeasible action explorations. The novelty and contributions of our work are two-fold:

1) We propose a RL solution incorporated with MCTS to tackle the MSOP. In this solution, the Q-learning with function approximation is employed as the basic learning architecture that allows multistep bootstrapping and continuous policy learning. To alleviate the computation burden of randomized multistep simulations, a MCTS algorithm is developed to
efficiently estimate the expected maximum action values in the iterative learning process.

2) We develop a knowledge incorporation scheme to embed the rules into the learning process. In this scheme, the probabilistic soft logic is adopted to map knowledge rules into potential functions. The potential functions are then combined by soft logic operations to confine the state-wise feasible action space and enhance the performance of the learned policy.

As far as we know, this is the first work of incorporating MCTS and domain knowledge into RL methods in power system applications.

The remainder of the paper is organized as follows. The problem is formulated in Section II. Its solution is given in Section III. The results in case studies are reported in Section IV and the paper concludes in Section V.

II. PROBLEM FORMULATION

Figure 1 presents a simplified configuration of the problem. The microgrid is connected to the distribution network at the point of common coupling (PCC). Components of the microgrid include DG, EV, other loads, and the BESS. The active power of these components is marked with a positive power flow direction in the figure. For notational convenience, we introduce $P_{SUM}$ to represent $(P_{DG} - P_{EV} - P_{CL})$.

![Fig. 1. Simplified architecture of a grid-connected microgrid.](image)

A. Constraints

Let $t$ be the time index. The active power constraint imposed by the distribution utility at the PCC is

$$ P_{PCC}^{min} \leq P_{PCC,t} = (P_{SUM,t} + P_{B,t}) \leq P_{PCC,t}^{max} $$

(1)

The branch power flow model developed in [25] is adopted for the power flow calculation of the microgrid,

$$ P_i^d + P_i^r = \frac{(P_i^d)^2 + (Q_i^d)^2}{U_i^2} = \sum_{k \in e(i,j)} P_{k,j}^d + P_{j,i}^d $$

(2)

$$ Q_i^d + Q_i^r = \frac{(P_i^d)^2 + (Q_i^d)^2}{U_i^2} = \sum_{k \in e(i,j)} Q_{k,j}^d + Q_{j,i}^d $$

(3)

$$ U_j^r = U_j^2 - 2U_j^2 P_i^r + (r_i^d)^2 + [(r_j^d)^2 + (x_j^d)^2] \frac{(P_i^d)^2 + (Q_i^d)^2}{U_i^2} $$

(4)

where the microgrid connection is denoted by a tuple $(l, i, j)$, namely line $l$ connecting hub bus $i$ and tail bus $j$. $P_i^d$ and $Q_i^d$ are active and reactive power delivered through the line. $U_i$ and $U_j$ are voltage magnitudes. $r_i^d$ and $x_j^d$ are resistance and reactance. $P_i^d$ and $Q_i^d$ are active and reactive power generation at bus $i$. $P_j^d$ and $Q_j^d$ are active and reactive power demand at bus $j$. $c(j)$ is the set of child buses of bus $j$. The left-hand side of (2)/(3) denote the active/reactive power injected into bus $j$. The right-hand side of (2)/(3) denote active/reactive power withdrawn from bus $j$.

The operational constraints in microgrid are given by (5)-(8),

$$ U_i^{min} \leq U_i \leq U_i^{max} $$

(5)

$$ (P_i^d)^2 + (Q_i^d)^2 \leq (S_i^{max})^2 $$

(6)

$$ P_{DG,j}^{min} \leq P_{DG,j} \leq P_{DG,j}^{max} $$

(7)

$$ P_{B,t}^{min} \leq P_{B,t} \leq P_{B,t}^{max} $$

(8)

where the node voltage amplitude, power flow in lines, power generation of DGs, and power charging/discharging of BESSs are constrained by their thresholds.

The $SOC$ of the BESS is given by

$$ SOC_t = SOC_{t-1}(1 - \sigma_t) - \eta_{B} \frac{P_{B,t} T}{CaV_{B,nom}} $$

(9)

In the charging mode, $\eta_{B} < 1$ and $P_{B,t} < 0$. In the discharging mode, $\eta_{B} > 1$ and $P_{B,t} > 0$.

To prevent damages caused by overcharge/overdischarge, the $SOC_t$ is restricted by

$$ SOC_{min} \leq SOC_t \leq SOC_{max} $$

(10)

The life-cycle throughput of a BESS is related to the number of operation cycles, $SOC$ in individual cycles, etc. [10, 22]

$$ Q_{life,t} = (N_{t} \times e^{-kSOC}) \times (1 - SOC_{t}). \frac{C_{B,nom, V_{B,nom}}}{1000 \text{ W} / \text{kW}} $$

(11)

where $k$ is an empirical parameter. The level of BESS degradation is measured by $|P_{B,t} M|/2Q_{life} [22-24]$.

B. Objective function

Assume $t$ is the decision time (which means all state variables are known up to $t$). In order to maximize the operational profits of the BESS, we can formulate the following multiperiod stochastic optimization model,

$$ z = \max_{a_1, a_2, \ldots, a_T} (\gamma E_{t+1} \max R(s_{t+1}, a_{t+1}) + \gamma E_{t+2} [\max R(s_{t+2}, a_{t+2})] + \ldots) $$

(12)

where $R(s_{t+1}, a_{t+1})$ is the immediate reward of taking action $a_{t+1}$ in state $s_{t+1}$. $P_{t+i|[t+i]}$ is the probability for the state transitions from $t+i$ to $t+i+1$. $\gamma$ is the discount factor.

In our case, $a_i$ and $s_i$ are given by

$$ a_i = P_{B,t} $$

(13)

$$ s_{i+1} = (SOC_{t}, P_{SUM,t}, c_t, P_{PCC,t}^{out}) $$

where $c_t$ and $P_{PCC,t}^{out}$ are time-of-use tariff and the active power at PCC expected by the distribution utility. The BESS is considered as the only dispatchable component. Because $s_{t+i}$ is an unknown future state at time $t$, $P_{SUM,t+i}$ is modeled as a stochastic variable mainly owing to the volatile energy resources. Here the superscript $\ast$ is used to distinguish stochastic variables from deterministic variables. This notation is also used hereinafter for vectors containing stochastic variables, such as $s_{t+i}$. Because our focus of uncertainty is the power generation/consumption in volatile energy resources, $c_{t+i}$ and $P_{PCC,t+i}^{out}$ are modeled as deterministic variables.

The immediate reward, $R_t$, contains three factors, defined as

$$ R_t = \alpha_1 R_{P,t} + \alpha_2 R_{cl,t} + \alpha_3 R_{B,t} $$

(14)

where $w_i$ ($i=1,2,3$) are the weights of the different factors. Individual factors are specified as
To allow continuous policy learning, the function approximation is employed in the above tabular Q-learning architecture that achieves a parametric approximation of the action value function, i.e. $Q(s, a; \theta) = \mathbb{E}(Q(s, a))$, where $\theta \in \Theta^d$ is a finite-dimensional weight vector. In this work, the basic neural network [26] is adopted as the function approximator. Its weights are updated following the gradient descent law, i.e.

$$\theta \leftarrow \theta + \alpha [Q(s, a; \theta) - Q(s, a; \bar{\theta})]\nabla Q(s, a; \theta).$$

The formulation of (20) distinguishes our Q-learning model from [11, 12] that do not incorporate the mechanism of multistep bootstrapping under uncertainty. However, this formulation increases the computation complexity as more simulation depths and scenarios need to be addressed for estimating the stochastically optimal rewards. We tackle this issue by customizing a MCTS algorithm in subsection B.

B. MCTS algorithm

Different from the MCTS algorithms developed in deterministic games, the MCTS in this work needs to incorporate stochastic scenarios into the estimation procedure of expected maximum action values. Here we outline the key ideas of the developed algorithm. More details of this algorithm is explained in the Appendix.

At decision time $t$, let $[\tilde{\omega}^{n}]$ be a stochastic vector for the probability distribution of stochastic variables over a planning horizon $n$, we refer to a scenario $\omega^{n} \in \Omega = \Omega_{s_{1}} \times \cdots \times \Omega_{s_{n}}$ as a realization (or sampling trajectory) of the stochastic process $[\tilde{\omega}^{n}]$. Then, we use the notion of SSP as a scenario sampling pool for providing the generative scenarios,

$$SSP = \{[\hat{P}^{ssp}_{SUM, j+1}, \hat{P}^{prop}_{SUM, j+1}], \ldots, [\hat{P}^{ssp}_{SUM, j+n}, \hat{P}^{prop}_{SUM, j+n}]\} \quad (21)$$

where $\hat{P}^{ssp}_{SUM}, \hat{P}^{prop}_{SUM}$ are lower and upper bounds for the confidence interval of $\hat{P}_{SUM}$.

From SSP, the generative scenarios containing $n$ stochastic variables $\{P_{SUM, j+1}, \ldots, P_{SUM, j+n}\}$ are sequentially sampled. Let $SSP^{n} = \{P^{n}_{SUM, j+1}, \ldots, P^{n}_{SUM, j+n}\}$ be the $n$th generative scenario, then a search tree successively expands from a father node $s^{f}_{i}$ to its child node $s^{c}_{i}$ (the father node is the node that is currently estimated). The tree is expanded based on the UCT (upper confidence bound for trees) policy

$$\arg\max_{s_{h} = \text{children of } s_{i}} \left( \frac{G^{n}_{i}(s_{h})}{N(s_{h})} + \beta \sqrt{\frac{\ln N(s_{h})}{N(s_{h})}} \right) \quad (22)$$

where $N(s_{i})$ and $N(s_{i+1})$ are the visit counts of the father and child node. $\beta$ is a constant variable determining the level of exploration. Initially, (22) prefers nodes with low visit counts. Asymptotically, the nodes that are promising with high values are identified. (22) balances the exploitation of learned value function and the exploration of unvisited nodes.

When the child node is selected in the $n$th scenario, the Monte Carlo rollout policy $\pi$, begins at this node and ends at the terminal node $s^{\tau}_{i}$. Each rollout performs a sequential simulation and constitutes $n$ state variables, we use $\{s^{n}_{i+1}, \ldots, s^{n}_{i+n}\}$ to denote the simulation trajectory in the $i$th rollout. Then the rollout statistics of all traversed edges are backed up.
where \( \mathbb{I} \) is the indicator function. If edge \( (s_{i+1}^n, a_{i+1}) \) was traversed, \( \mathbb{I}(s_{i+1}^n, a_{i+1}) = 1 \); Otherwise \( \mathbb{I}(s_{i+1}^n, a_{i+1}) = 0 \). \( G_{i+1}^{i\alpha} \) is the accumulated reward from the node \( s_{i+1} \) to the end node \( s_t \).

Equations (23)-(24) updates the visit counts and mean action value function in all simulations passing through that edge. We then identify the set of best-performing actions and obtain the n-stage maximum action value for \((s_t, a_t)\) after \(L \) rollouts are executed in the \(m\)th scenario,

\[
G_{i+1}^{i\alpha}(s_t, a_t) = R_{i+1}(s_t, a_t, s_{i+1}) + \gamma \max_{a_{i+1}} G_{i+1}^{i\alpha}(s_{i+1}, a_{i+1} | SSP^\alpha) \tag{25}
\]

By repeating the above process, the expected maximum action value is approximated as

\[
\max \mathbb{E} G_{i+1}^{i\alpha}(s_t, a_t) \approx M \max_{a_{i+1}} \mathbb{E} G_{i+1}^{i\alpha}(s_{i+1}, a_1 \ldots a_i) \tag{26}
\]

where \( M \) is the number of scenarios investigated.

There are two differences that distinguish the above MCTS and the MCTS deployed in Go [15, 16]. The first one is that in Go only one deterministic scenario is investigated for estimating the value function. In our case, we incorporate different possible scenarios for deriving the expected value function. This is achieved by using the notion of SSP in (21) to allow scenarios generation based on any explicit or implicit probability function, and the expected optimal value are accumulated from individual scenarios by (23)-(24). The second difference is that in Go only the estimated value of the last-stage state (i.e. the terminal node) in each rollout is backed up for updating the value function, which is not an accurate estimation in our case. Thereby, we temporally memorize and accumulate the action values of each transition between father and child nodes by (25)-(26) for updating the value function in each rollout.

### C. Scheme for incorporating knowledge rules

Two definitions are given below to leverage different rules for reducing infeasible explorations in the RL algorithm.

**Definition 1.** Let \( \mathcal{K} = \{\sigma_i, k_i(a | s)\}_{i=1}^n \) be a set of weighted rule sets, where \( k_i(a | s) \) is the \(i\)th rule estimating the feasibility of action \(a\) conditioned on state \(s\), \( \sigma_i \) is the weight of \(k_i\).

In practice, the knowledge rules can be classified as hard rules and soft rules. Here we consider three rules in the rule set,

\[
k_1(a | s_i, s_i+1) : \text{soc}^\text{emp} \leq \text{soc}^\text{emp}_{i+1} \leq \text{soc}^\text{emp}
\]

\[
k_2(a | s_i, s_i+1) : P_{\text{PCC}}^\text{emp} \leq P_{\text{PCC}}^\text{emp}_{i+1} \leq P_{\text{PCC}}^\text{emp}
\]

\[
k_3(a | s_i, s_i+1) : |P_{\text{PCC}}^\text{emp}_{i+1} - P_{\text{PCC}}^\text{emp}| \leq P_{\text{Threshold}}^\text{emp}
\]

where \(k_1\) and \(k_2\) are hard rules that require \(\text{soc}^\text{emp}\) and \(P_{\text{PCC}}^\text{emp}\) to remain within allowable ranges when taking action \(a_i\) in state \(s_i\) and transitioned to a successor state \(s_{i+1}\). The hard rules are definitely not violated, otherwise the security of power distribution network or the BESS will be damaged. \(k_3\) is a soft rule that expects the actual \(P_{\text{PCC}}^\text{emp}\) to have small fluctuations between successive states when taking an action. How to use a soft rule depends on actual needs. For example, when the BESS is funded by an end user who focuses only on electricity revenue, \(k_3\) can be relaxed because otherwise some candidate actions with higher rewards will be excluded.

**Definition 2.** Let \( \phi_i(a | s) \) be an individual potential function of action \(a\) conditioned on state \(s\) and examined by rule \(k_i\). Let \( \Phi_i(a | s) \) be the total potential function of action \(a\) conditioned on \(s\) and examined by the rule set \( \mathcal{K} \). Also, let \( \mathcal{A}_j \) be the set of feasible action spaces evaluated by \(\Phi_j(a | s)\).

\(\phi_i(a | s)\) can be seen as the numerical expression of rule \(k_i\). However, when there exists multiple rules, the logic inferences among them are needed for deriving a final result of the feasibility of candidate actions, especially when these rules are not consistent in evaluating the feasibility of an action. Therefore, we introduce PSL to map knowledge rules into the scalar values taken in the interval \([0, 1]\). The mapping of \(k_i\) into an individual potential function is typically of the form \(\phi_i = (\max(0,d_i))\), where \(d_i\) is a measure of the distance to satisfiability of \(k_i\) [19, 20]. For hard rules \(k_i\) \((l=1, 2)\), \(d_i = 1\) when the candidate action is evaluated as feasible according to \(k_i\); otherwise \(d_i = 0\). For the soft rule \(k_3\), an exponential operator is used to measure its distance to satisfiability, i.e., \(d_i = \exp(-\frac{|P_{\text{PCC}}^\text{emp}_{i+1} - P_{\text{PCC}}^\text{emp}|}{P_{\text{Threshold}}^\text{emp}})\).

We then derive the total potential function \(\Phi_j(a | s)\) from all individual potential functions using certain logic operators. Because we have soft rules that take truth values in \([0, 1]\), the classic Boolean logic is replaced by the Lukasiewicz logic that allows continuous truth values taken from the interval \([0, 1]\). The logic operators such as AND (\(\land\), OR (\(\lor\), NOT (\(\neg\) are redefined as [19, 20]

\[
\begin{align*}
\phi_i \land \phi_j & = \max\{\phi_i + \phi_j - 1, 0\} \\
\phi_i \lor \phi_j & = \max\{\phi_i + \phi_j - 1, 0\} \\
\neg \phi_i & = 1 - \phi_i
\end{align*}
\]

This redefinition allows a simple and flexible inference among different rules. In this work, let \(\bar{\phi}\) be the total potential function of all hard rules, and \(\hat{\phi}\) be the potential function of all soft rules, we have \(\bar{\phi} = \phi_1 \land \phi_2 \ldots \phi_3 = \phi_1 \land \phi_3\). Hence \(\mathcal{A}_j\) is decided by

\[
\mathcal{A}_j = \{a_j \in \mathbb{A} | \Phi_j(a_j | s) \geq \sigma_j\}
\]

where \(\sigma_j\) is the threshold.
D. The developed RL algorithm

Fig. 2 Flow chart of the proposed RL algorithm.

Fig. 3 Microgrid system.

Fig. 4 Hourly active power of the microgrid.

IV. CASE STUDY

In this section, case studies were conducted to test the proposed algorithm. Figure 3 presents the microgrid system tested [23], which contains two PVs, two EV charging piles, one BESS, and other loads connected to each node. The BESS was a 500 kWh lead-acid battery. Figure 4 depicts the hourly active power of different components in the microgrid, which showed the high volatility of DGs and EVs. In the stochastic scenarios, we assumed the standard deviations of the forecast errors for stochastic variables as 5%. For simplicity, $P_{\text{PCC}}$ was set to a constant value of 50 kW, and the TOU tariffs were referenced from the actual tariffs in China. For the thresholds of the knowledge rules, we restricted the $\text{SOC}$ in rule $k_1$ to be within [30%, 90%], $P_{\text{PCC}}$ in rule $k_2$ was set to [0, 100 kW], and the variation between the $P_{\text{PCC}}$ of two consecutive states in rule $k_3$ was maintained below 50 kW. The training and testing procedures of our algorithm followed [11, 28]. The parameters $\varepsilon$ in (18) and $\beta$ in (22) were set to 1% and 0.7, respectively. The bootstrapping stage was set to 4.

We first tested the feasibility of the proposed algorithm in realizing its objectives expressed in (12). Figure 5 shows the power management results of BESS in nearly three consecutive days. In sub-figure (a), $P_{\text{SUM}}$ fluctuated significantly because of the volatile resources DGs and EVs. In contrast, the dispatching of BESS regulated $P_{\text{PCC}}$ for a close tracking of $P_{\text{PCC}}$. Sub-figure (b) shows that the dispatching solution of BESS in general procures energy during low-price low-load periods and sells energy during high-price high-load periods, which increased the electricity revenues. Moreover, a regular charging/discharging behavior of BESS is showed by the $\text{SOC}$ curve, that can prevent accelerating the degradation from over-charging / over-discharging.

We then analyzed the computation performance of the developed MCTS algorithm, whose role is mainly to give efficient estimations of the maximum action values over multistep simulation trajectories. To evaluate the degree of accomplishment of this role, we compared the numerical results of MCTS and three algorithms by varying the number of iterations while fixing the simulation scenarios. As listed in Table I, the compared algorithms include a random search algorithm (RS) that used a random policy during bootstrapping, an exhaustive search algorithm (ES) enumerating candidate
actions, and a heuristic search algorithm based on the genetic algorithm (GA). The number of iterations in the numerical tests were varied from $10^1$ to $10^4$. In each iteration budget, we repeated the computations of these algorithms for 10 times and recorded the mean and variance of different algorithms. The mean values were normalized by the min-max normalization. The variances of BS were omitted because they were all zero.

### Table I

**Performances of Different Algorithms in Estimating the Maximum Action Values**

| Number of iterations | Mean of maximum action value | Variance of estimation |
|----------------------|-----------------------------|------------------------|
|                      | MCTS | RS | GA | ES | MCTS | RS | GA |
| $10^1$               | 0.81 | 0.34 | 0.17 | 0 | 0.62 | 0.99 | 0.49 |
| $10^2$               | 0.92 | 0.42 | 0.26 | 0.18 | 0.34 | 1.56 | 0.11 |
| $10^3$               | 0.97 | 0.38 | 0.34 | 0.42 | 0.10 | 2.00 | 0.01 |
| $10^4$               | 1    | 0.39 | 0.38 | 0.74 | 0.02 | 2.17 | 0    |

The mean value indicates that MCTS is the most efficient algorithm in discovering the maximum action values. The variance of MCTS is asymptotically reduced as the number of iterations increased, which justifies the robustness and asymptotic convergence of this algorithm. However, ES is the least efficient in estimating the action values. RS is highly stochastic without convergence guarantees regardless of the increase of iterations. For GA, although its variance is the smallest and reached almost 0 after $10^4$ iterations, its estimations of the maximum action value improves slowly when the computation effort increases. One possible explanation is that the iterative searching in GA is stuck in some local optimum after $10^4$ iterations. From above comparison, we can conclude that the MCTS is the best-performing algorithm that balances the accuracy of local search and adaptability towards the global optimum.

We further demonstrated the performance of incorporating the knowledge rules into supervising the Q-learning process. In this test, we compared the proposed algorithm (knowledge incorporation, 4-stage bootstrapping) with two other algorithms, namely Algorithm 1 (no knowledge incorporation, 4-stage bootstrapping) and Algorithm 2 (knowledge incorporation, 1-stage bootstrapping) in terms of the rewards obtained and the actual dispatching results. To increase the learning efficiency when extending the bootstrapping stages from 1 to 4, in our algorithm the immediate reward of one-step states transition was set as the initial value for the follow-up action value updating. Figure 6 depicts the accumulated rewards obtained by these algorithms along the learning trajectory. It shows our algorithm is the most effective one in maximizing the cumulative rewards. Specifically, the advantages in the estimating rewards of our algorithm over Algorithm 1 and 2 are highlighted in the earlier and later learning stage, respectively. This result shows that knowledge incorporation is effective when the agent has little exploration experiences. Moreover, extending the bootstrapping depth in conjunction with knowledge incorporation can facilitate the agent to increase its rewards in the long run.

![Fig. 6 Comparison of accumulated rewards obtained by three algorithms.](image)

**Table II**

**Dispatch Results of the Proposed Algorithm and Algorithm 2**

|                           | Proposed algorithm | Algorithm 2 |
|---------------------------|--------------------|-------------|
| Electricity revenue       | 1.05               | 1           |
| BESS degradation cost     | 0.92               | 1           |
| Standard deviation of $P_{PCC}$ | 0.79       | 1           |

**V. Conclusion**

In this paper, we present a MSOP for the dynamic management of BESSs in microgrids. The model is developed to minimize the operational costs of the microgrid, taking into account the nonconvex degradation cost function of the BESS. Then, we provide a RL solution augmented with MCTS and
knowledge rules. We first express the knowledge rules into the potential function in the form of soft logic. These knowledge rules are used to confine the state-wise action space, which can reduce the number of infeasible actions explored by the RL agent. To alleviate the computation burden of multistep bootstrapping under uncertainty, a MCTS algorithm is customized to increase the estimation efficiency of the expected maximum action values. The results of our numerical tests show that the proposed algorithm asymptotically optimizes the dispatch policy and outperforms other algorithms.

APPENDIX

The MCTS procedure mainly composes of five steps as shown in Fig. 9.

a. Generation. This stage generates randomized sequences containing $n$ sequential stochastic variables, i.e.

\[
\begin{align*}
&\text{a. Generation} \\
&\quad \bullet \quad \dot{s}, \dot{a}, \ldots
\end{align*}
\]

b. Selection. This stage selects exploitative policies in the generated scenario. Assume the current in-tree simulation phase begins at node $\dot{s}_{i+1}$ and ends at $\dot{s}_{i,n}$, each node $\dot{s}$ of the tree contains edges $(s,a)$, and each edge stores a set of statistics $(G(s,a), N(s,a))$, where $N(s,a)$ is the visit count and $G(s,a)$ is the mean action value for that edge.

c. Expansion. This stage incrementally expands the tree until the terminal nodes in a generative scenario. The UCT criterion is used to decide which child node to be expanded. Then the Monte Carlo rollout policy $\pi$ begins at this node and ends at a terminal node. During tree expansion, the successively joined leaf nodes result in different combinations of sequential state-action pairs.

d. BackPropagation. This stage updates the rollout statistics of each in-tree node backwards from the terminal node to the root node by (23) and (24). After reaching computation budget, the set of state-action pairs with the highest expected rewards is identified as marked in the red rectangle in Fig. 9.

e. Update. This stage updates the action value estimation results for each scenario by (25), and finally accumulate expected action value estimations of all scenarios by (26).

REFERENCES

[1] M. D. Hopkins, A. Pahwa, and T. Easton, “Intelligent dispatch for distributed renewable resources,” IEEE Trans. Smart Grid, vol. 3, no. 2, pp. 1047-1054, Jun. 2012.
[2] H. Karami, M. Sanjari, S. H. Hosseinian, et al, “An optimal dispatch algorithm for managing residential distributed energy resources,” IEEE Trans. Smart Grid, vol. 5, no. 5, pp. 2360-2367, Sep. 2014.
[3] M. Mahmoodi, P. Shamsi, and B. Fahimi, “Economic dispatch of a hybrid microgrid with distributed energy storage,” IEEE Trans. Smart Grid, vol. 6, no. 6, pp. 2607-2614, Nov. 2015.
[4] A. D. Giorgio, F. Liberati, A. Lanna, et al, “Model Predictive Control of Energy Storage Systems for Power Tracking and Shaving in Distribution Grids,” IEEE Trans. on Sustain. Energy, vol. 8, no. 99, pp. 496-504, 2017.
[5] S. Talari, M. Yazdaninejad, and M. R. Haghifam, “Stochastic-based scheduling of the microgrid operation including wind turbines, photovoltaic cells, energy storages and responsive loads,” IET Gener. Transm. Distrib., vol. 9, no. 12, pp. 1498-1509, Sep. 2015.
[6] X. Xi, R. Sioshanshi, and V. Marano, “A stochastic dynamic programming model for co-optimization of distributed energy storage,” Energy Syst., vol. 5, no. 3, pp. 475-505, 2014.
[7] W. Su, J. Wang, and J. Roh, “Stochastic energy scheduling in microgrids with intermittent renewable energy resources,” IEEE Trans. Smart Grid, vol. 5, no. 4, pp. 1876-1883, Jul. 2014.
[8] T. A. Nguyen and M. L. Crow, “Stochastic Optimization of Renewable-Based Microgrid Operation Incorporating Battery Operating Cost,” IEEE Trans. Power Syst., vol. 31, no. 3, pp. 2289-2296, May 2016.
[9] A. Bhatchayara, J. P. Kraroufeh and B. Zeng, “Managing Energy Storage in Microgrids: A Multistage Stochastic Programming Approach,” IEEE Trans. Smart Grid, vol. 9, no. 1, pp. 483-496, Jan. 2018.
[10]C. Liu, X. Wang, X. Wu, et al, “Economic scheduling model of microgrid considering the lifetime of batteries,” IET Gener. Transm. Distrib., vol. 11, no. 3, pp. 759-767, 2017.
[11]W. Liu, et al, “Distributed Economic Dispatch in Microgrids Based on Cooperative Reinforcement Learning,” IEEE Trans. Neural Networks & Learning Syst., vol. 29, no. 6, pp. 2192-2203, 2018.
[12]Q. Wei, D. Liu, G. Shi, “A novel dual iterative Q-learning method for optimal battery management in smart residential environments,” IEEE Trans. Ind. Electron., vol. 62, no. 4, pp. 2509-2518, 2015.
[13]R. S. Sutton, A. G. Barto, “Reinforcement Learning: An Introduction,” MIT Press, London, 2018.
[14]R. Couloum, “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search,” in Proc. 5th Int. Conf. Comput. and Games, Turin, Italy, 2006, pp. 72-83.
9

[15] C. Browne, E. Powley, D. Whitehouse, et al, “A Survey of Monte Carlo Tree Search Methods”, IEEE Trans. Comput. Intel. and AI in Games, vol. 4, no. 1, pp. 1–49, 2012.
[16] D. Silver, A. Huang, et al, “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484–489, 2016.
[17] P. Christiano, J. Leike, T. B. Brown, et al, “Deep reinforcement learning from human preferences,” In the Annual Conference on Neural Information Processing Systems (NeurIPS), 2017.
[18] J. Huang, F. Wu, D. Precup, et al, “Learning Safe Policies with Expert Guidance,” in 32nd Conference on Neural Information Processing Systems (NeurIPS), Montréal, Canada, 2018.
[19] S. Bach, M. Broecheler, B. Huang, et al, “Hinge-Loss Markov Random Fields and Probabilistic Soft Logic,” Journal of Machine Learning Research, vol. 18, pp. 1–67, 2017.
[20] Z. Hu, X. Ma, Z. Liu, et al, “Harnessing Deep Neural Networks with Logic Rules”, in Proc. 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016, pp. 2410-2420.
[21] M. Sachan, A. Dubey, T. Mitchell, et al, “Learning Pipelines with Limited Data and Domain Knowledge: A Study in Parsing Physics Problems”, in 32nd Conference on Neural Information Processing Systems (NeurIPS), Montréal, Canada, 2018.
[22] Lambert T, Gilman P, Lilienthal P. Micropower system modeling with homer, 2006. www.homerenergy.com/documents/MicropowerSystemModelingWithHOMER.pdf.
[23] T. Ma, H. Yang, L. Lu, “A feasibility study of a stand-alone hybrid solar-wind-battery system for a remote island,” Applied Energy, vol. 121, pp. 149-158, 2014.
[24] Zhao, B., Zhang, X., Chen, J., et al.: ‘Operation optimization of standalone microgrids considering lifetime characteristics of battery energy storage system’, IEEE Trans. Sustain. Energy, 2013, 4, (4), pp. 934-943.
[25] M. E. Baran and F. F. Wu, “Optimal capacitor placement on radial distribution systems,” IEEE Trans. Power Del., vol. 4, no. 1, pp. 725-734, Jan. 1989.
[26] J. Si, Y. Wang, “On-line learning control by association and reinforcement,” IEEE Trans. Neural Netw., vol. 12, no. 2, pp. 264-276, 2001.
[27] G. Antonis, D. Nikos, “Centralized Control for Optimizing Microgrids Operation,” IEEE Trans. Energy Conversion, vol. 23, no. 1, pp. 241-248, 2008.
[28] R. Rocchetta, L. Bellani, M. Compare, et al, “A reinforcement learning framework for optimal operation and maintenance of power grids,” Applied Energy, vol. 241, pp. 291-301, 2019.