Active Power Correction Strategies Based on Deep Reinforcement Learning——Part II: A Distributed Solution for Adaptability

Siyuan Chen, *Student Member*, *IEEE*, Jiajun Duan, *Member*, *IEEE*, Yuyang Bai, Jun Zhang, *Senior Member*, *IEEE*, Di Shi, *Senior Member*, *IEEE*, Zhiwei Wang, *Senior Member*, *IEEE*, Xuzhu Dong, *Senior Member*, *IEEE*, Yuanzhang Sun, *Senior Member*, *IEEE*

***Abstract*—This article is the second part of Active Power Correction Strategies Based on Deep Reinforcement Learning. In Part II, we consider the scenarios of renewable energies plugged in the large-scale power grid and provide an adaptive algorithmic implementation to maintain power grid stability. On the basis of the robustness method in Part I, a distributed deep reinforcement learning method is proposed to improve the capability of overcoming the influence of the increasing renewable energy penetration. A multi-agents system is implemented in multiple control areas of the power system, which conducts a fully cooperative stochastic game. Based on the Monte Carlo tree search mentioned in Part I, we select effective actions in each sub-control area to search the Nash equilibrium of the game. Based on the QMIX method, a structure of offline centralized training and online distributed execution is proposed to employ the better effective actions in the active power correction control. Our proposed method is evaluated in the modified global competition scenario cases of "2020 Learning to Run a Power Network - Neurips Track 2".**

***Index Terms*— active power correction strategies, renewable energies, distributed deep reinforcement learning, stochastic game, Nash equilibrium.**

SYMBOLS

Functions

$C_{loss}(\bullet)$	function of power loss cost
$C_{redis}(\bullet)$	function of generation redispatch cost
$C_{bkt}(\bullet)$	function of blackout cost
$R(\bullet)$	function of cumulative reward
$r_{i,t}(\bullet)$	immediate reward function of agent *i* at time *t*
$g_t(\bullet)$	performance function of agent *i* at time *t*
$V_i(\bullet)$	function of state value
$p(\bullet)$	function of state transition probability
$Q_i(\bullet)$	Q function of agent *i*
$Q_{tot}(\bullet)$	joint Q function of multi-agents.
$L(\bullet)$	loss function of Q network

Variables

T_e	duration of the blackout
r_l	resistance of line *l*
y_l	active power of line *l*
$P_{G_{i,t}}$	active power output of generator *i* at time slot *t*
$P_{j,t}$	energy loss in the presence of blackout
H_l	the binary variable of BBS action
$\pi_{i,k}$	strategy of agent *i*
$\theta, \hat{\theta}$	parameters of Q network
τ_i	action-observation history of agent *i*
$\mathbf{a}_k$	joint action of multi-agents
$\boldsymbol{\pi}_k$	joint strategy of multi-agents
$\boldsymbol{\pi}_k^*$	optimal joint strategy of multi-agents
$\boldsymbol{\tau}$	joint action-observation history of multi-agents
$\boldsymbol{\theta}$	network parameters of multi-agents

Sets

$\mathbf{N}_l$	set of lines
$\mathbf{N}_{Gen}$	set of generators
$\mathbf{N}_{Load}$	set of loads
$\mathbf{A}_{ag}$	set of multi-agents
$\mathbf{S}$	state space of multi-agents
$\mathbf{O}_i$	partial observation space of agent *i*
$\mathbf{A}_i$	actions space of agent *i*
$\mathbf{T}$	operation duration of power system

Constants

$N_l^{\max}$	maximum number of allowable lines switching
ρ_l	usage of lines *l* capacity
φ	penalty coefficient of heavy load
ψ	penalty coefficient of overload
κ	penalty term of blackout in reward function

This work was supported by the National Key R&D Program of China under Grant 2018AAA0101502. *(Corresponding author: Jun Zhang.)*

S. Chen, Y. Bai, J. Zhang, X. Dong and Y. Sun are with the School of Elec trical Engineering and Automation, Wuhan University, Wuhan 430072, China (e-mail: wddqcsy@whu.edu.cn; baiyuyang@whu.edu.cn; jun.zhang.ee@whu.edu.cn; dongxz@whu.edu.cn; yzsun@mail.tsinghua.edu.cn).

J. Duan, D. Shi and Z. Wang, are with GEIRI North America, San Jose, C A 95134, USA. (e-mail: jiajun.duan@geirina.net; di.shi@geirina.net; zhiwei.wang@geirina.net).

DOI: 10.17775/CSEEJPES.2020.07070

ζ	coefficient of soft update
ε	greedy parameter

I. Introduction

Large-scale plug-in distributed renewable energy resources (RESs) are gradually becoming a significant feature of a power grid. The randomness and volatility of RESs generation may lead to transmission lines' capacity overload when the power system has on-peak demand or maintenance schedule. Considering that the uncertainty of the power system will increase with the penetration rate of RESs, active power correction control (APCC) is an essential method to maintain the stability of the power system active power. For APCC strategies, a large amount of research focuses on generation redispatch, load shedding, or demand response. Among them, several studies have exploited a less costly method with great potentials, namely the grid topology reconfiguration. In [1], based on the real cases of the American power grid, simulations are conducted to analyze more than 1500 thousands kinds of faults, and results demonstrate that topology change, i.e., bus bar switching, transmission switching, can effectively eliminate or alleviate the transmission lines overload in most fault scenarios.

Currently, the expanding scale of the power grid and the wide applications of power electronic devices have continuously increased the high dimensionality and nonlinearity of the power grids. Power grids are increasingly difficult to model accurately using traditional mathematical and physical mechanisms, which provide an opportunity for the applications of artificial intelligence technology in dispatching and control problems of the power system. Deep reinforcement learning (DRL) is a combination of reinforcement learning (RL) and deep learning technologies, which learn directly from agents' interactions with an environment. In [2], a Q-learning-based method is proposed to conduct the optimal control of reactive voltage, which solves the convergence problem of traditional reactive power optimization for nonlinear integer programming models. In [3] and [4], a DRL method is applied in the scenarios of voltage instability, which constructed low-voltage load shedding strategies and shunt capacitor switching schemes. The effectiveness of the proposed method is verified through several unknown fault scenarios. In [5], a voltage control strategy based on DQN and DDPG algorithms are applied in automatic voltage control (AVC). It ensures the voltage of each bus stays within the standard range, which is based on the information collected by SCADA or PMUs. At present, DRL has gained extensive attention and remarkable achievements in the fields of games [6], medical treatments [7], autonomous driving [8], unmanned aerial vehicles [9], and so on. However, it is difficult for DRL to perform well in high-dimensional action space, i.e., generally more than 10^4 kinds of actions. "The curse of dimensionality" caused by the complexity of the power system becomes one of the biggest obstacles to the practical applications of DRL.

As an extension technology of DRL, distributed deep reinforcement learning (DDRL) has been applied in a multi-agent system (MAS) manner, and effectively solved the problem of high-dimensional action space faced by DRL [10]. DDRL deploys multiple agents on buses of the large-scale power grid and divides the power grid into multiple sub-control areas. Due to the interaction and collaboration between multiple agents, the dimension of each agent's action space can be reduced to an acceptable level. In [11], Didi Company has developed a multi-agent reinforcement learning framework based on semantic information, which can support both DQN and advantage actor-critic (A2C) algorithms to solve the problem of large-scale vehicle scheduling and management. In [12], combining independent Q-learning (IQL) with DQN, a DDRL framework is proposed to make each agent has an independent Q network. However, due to the fact that the IQL algorithm does not conduct interactions among multiple agents, as a result, the environment for each agent is unknown and non-static. This violates the principle of the Markov decision process (MDP) and cannot be proven to achieve the convergence of the algorithm. In [13], considering the dynamic instability of the environment caused by the IQL algorithm, a value-decomposition networks (VDN) algorithm is proposed to obtain the joint action-value function by summing the Q-value of each agent. Numerical results demonstrate that the VDN algorithm can improve the convergence of DDRL. In [14], based on VDN, a QMIX algorithm is proposed by constructing a mixing network to integrate local value functions. In the training process, global information is added to further improve the network's ability of fitting Q values. However, DDRL has not been studied and applied in the field of power systems.

In this paper, we propose a DDRL framework for joint control strategies of the APCC problem in large-scale power systems. Specifically, the APCC model with topology reconfiguration actions is established to formulate the basic problem. The fully cooperative stochastic game is then utilized to model the interactions between active power controllers (APC). A model-free model is adopted to search for the Nash equilibrium (NE) for the game. Considering that the control issue of the APCC problem is discrete, a QMIX method is adopted, which is modified from [14]. Based on the characteristic of the QMIX method, a structure of online centralized training and offline distributed executing is proposed to satisfy the practical application requirements of large-scale power systems. Our method is verified in an open-source platform with relevant scenarios and cases.

The rest of this paper is organized as follows: The formulation of the APCC problem and the stochastic game is described in Section II. The method based on DDRL used to search for the NE is illustrated in Section III. The performance of the proposed method is verified by case studies in Section IV. The conclusions and suggestions for future work are given in Section V.

II. Problem Formulation

A. APCC Model

Active power correction control of power systems is a non-linear, mixed, integer programming problem, which is normally solved by sensitivity methods and optimization programming methods. Generally, the APCC is achieved by generation redispatch and load shedding with limited effects on power flow control. Bus bar switching (BBS), which can switch the elements from a bus bar to another, is an effective means to correct power flow quickly and effectively [15]. Considering that the influence of RESs required rapid response to prevent branch power flow from being off-limit, this paper adopts the optimization programming method of BBS. The mathematic model of the correction control is as follows:

$$\min_{y_l, P_{G_{i,t}}} \sum_{t=1}^{t_{end}} \left(C_{loss}(t) + C_{redis}(t) \right) + \sum_{t=t_{end}}^{T_e} C_{bkt}(t) \tag{1}$$

$$C_{loss}(t) = \sum_{l \in \mathbf{N}_l} r_l y_l^2(t) \tag{2}$$

$$C_{redis}(t) = \alpha \sum_{i \in \mathbf{N}_{Gen}} \left| P_{G_{i,t-1}} - P_{G_{i,t}} \right| \tag{3}$$

$$C_{bkt}(t) = \sum_{j \in \mathbf{N}_{Load}} \beta P_{j,t} \tag{4}$$

where $C_{loss}(t)$, $C_{redis}(t)$ and $C_{bkt}(t)$ are power loss cost, generation redispatch cost and blackout cost, respectively. t_{end} is the time when the power system blackout. T_e is the duration of the blackout. $\mathbf{N}_l$, $\mathbf{N}_{Gen}$ and $\mathbf{N}_{Load}$ are set of lines, generators and loads, respectively. r_l and y_l are the resistance and active power of line *l*, respectively. $P_{G_{i,t}}$ is the active power output of generator *i* at time slot *t*. $P_{j,t}$ is the energy loss in the presence of blackout.

In this paper, we focus on applying the topology actions of BBS, which is less costly than generation redispatch. A binary variable H_l is used to denote BBS action, which is 1 when line *l* is switched. Considering the stability of the power system, the BBS action in a time should be restricted, which is

$$\sum_{l \in N_l} (1 - H_l) \le N_l^{\max} \tag{5}$$

where $N_l^{\max}$ is the maximum number of allowable lines switching.

B. Stochastic game and Nash equilibrium

Limited by the action space dimension and observation space size of a large-scale power system, a single agent is not able to handle all the functionalities and management schemes in practical applications. A multi-agents system is essential to deal with "the curse of dimensionality", which can implement a cooperative control and management scheme in large-scale power grids. In the framework of DDRL, each agent follows the basic learning paradigm of reinforcement learning, which needs to consider both its exploration and the impact of other agents' strategies on the environment. The interaction among agents can be captured in the form of a cooperative stochastic game. The main components of the game include:

- Agent: APCs in the set $\mathbf{A}_{ag}$;
- State: the states s_t of the power system include active power outputs of generators, usage of lines capacity, electrical quantities, etc.. $s_t \in \mathbf{S}$, where $\mathbf{S}$ is the state space;
- Observation: partial observation $o_{i,t}$ based on the functionality of agent *i*. $o_{i,t} \in \mathbf{O}_i$, where $\mathbf{O}_i$ is the partial observation space of agent *i*;
- Action: BBS actions $a_{i,t}$ of each agent or do nothing. $a_{i,t} \in \mathbf{A}_i$, where $\mathbf{A}_i$ is the action space of agent *i*;
- Reward: immediate reward $r_{i,t}$.

At the beginning of time slot $t \in \mathbf{T}$, where $\mathbf{T}$ is the operation duration, RESs will be connected randomly to the power grid, which may cause overloads of transmission lines. If the usage ratios of lines are controlled within a certain range, the power losses will be reduced and the power system is expected to survive longer. Hence, based on (1)-(4), the performance at time *t* can be formulated as [16]:

$$g_t = \sum_{l \in \mathbf{N}_l} \left(\begin{array}{l} \max\left(0, 1 - \rho_l^2\right) - \\ \varphi \cdot \max\left(0, \rho_l - 0.9\right) - \psi \cdot \max\left(0, \rho_l - 1\right) \end{array} \right) \tag{6}$$

where ρ_l is the usage of lines *l* capacity. φ and ψ are the penalty coefficient of heavy load and overload, respectively.

Each APC will obtain a cumulative reward when the power system maintains its normal operation and get much larger negative rewards if power system blackout happens. Thus, the APCs perform a cooperative stochastic game to achieve a Nash equilibrium based on their observations. The immediate reward and cumulative reward of APC *i* at time *k* can be formulated as

$$r_{i,t} = \begin{cases} \kappa & \textit{if blackout} \\ \sum_{i=0}^{t} g_t & \textit{otherwise} \end{cases} \tag{7}$$

$$R(s_k) = \sum_{t=k}^{\mathbf{T}} \gamma_i^{t-k} r_{i,t} \tag{8}$$

where κ is a negative constant. γ_i is the discount factor of APC *i*. s_k is the states at time k.

The APCs aim to maintain the normal operation of the power system, i.e., the foresighted reward. We can calculate the accumulative reward of all states unless the game is over. Thus, a value function is introduced to evaluate the potential future reward of states, which is:

$$\begin{aligned} V_i(s_k) &= \mathrm{E}\left[R(s_k) \middle| S_t = s_k \right] \\ &= \mathrm{E}\left[R(s_{k+1}) + \gamma v_i(s_k) \middle| S_t = s_k, A_t = \mathbf{a}_k \right] \\ &= \sum_{s_{k+1}, r_{i,k}} p\left(s_{k+1}, r_{i,k} \middle| s_k, \mathbf{a}_k\right) \left[r_{i,k} + \gamma V_i(s_{k+1}) \right] \end{aligned} \tag{9}$$

where $\mathbf{a}_k = \left[a_{1,k}, \ldots, a_{i,k}, \ldots, a_{N,k} \right]^T$ is the joint action, $p\left(s_{k+1}, r_{i,k} \middle| s_k, \mathbf{a}_k\right)$ is state transition probability, that is, $p : s_k \times \mathbf{a}_k \times s_{k+1} \to [0,1]$. (9) demonstrates that the stochastic game has Markov properties. $\pi_{i,k} : s_k \to \mathbf{a}_{i,k}$ denotes the strategy of APC *i*, and the joint strategy of the APCs can be described as $\boldsymbol{\pi}_k = \left[\pi_{1,k}, \ldots, \pi_{i,k}, .., \pi_{N,k} \right]^T$. The value function is related to APCs' strategies, which can be denoted by $V_i(s_k, \boldsymbol{\pi}_k)$.

The solution of the stochastic game is NE, which is a state of the game where no agent can benefit by unilaterally changing strategies. Assumed that the NE solution of APCC problem denotes by $\boldsymbol{\pi}_k^* = \left[\pi_{1,k}^*,...,\pi_{i,k}^*,...,\pi_{N,k}^*\right]^T$, the optimality of the NE solution can be described as:

$$V_i\left(s_k, \boldsymbol{\pi}_k^*\right) \geq V_i\left(s_k, \boldsymbol{\pi}_{k,-i}^*\right), \forall i \in N \tag{10}$$

where $\boldsymbol{\pi}_{k,-i}^* = \left[\pi_{1,k}^*,...,\pi_{i,k},...,\pi_{N,k}^*\right]^T$ are the optimal strategies of APCs excluding the APC *i*.

In the stochastic game of APCC, we solve the optimization problem of Q function to obtain the NE solution based on the Bellman equation, which is:

$$\max_{\boldsymbol{\pi}_k} Q_i\ \left(s_k, \mathbf{a}_k\right) = \sum_{\mathbf{s}_{k+1} \in \mathbf{S}} p\left(s_{k+1}, r_{i,k} \left| s_k, \mathbf{a}_k\right.\right)\left[r_{i,k} + \gamma V_i\left(s_{k+1}\right)\right] \tag{11}$$

Therefore, we need to search the maximum Q value to obtain the NE solution $\boldsymbol{\pi}_k^*$. To address this issue, in previous literature some model-based methods attempt to solve for the actions of APCs, however, the performance of these methods depends on the accuracy of the models. Hence, a model-free method, i.e., DQN, is adopted in this paper to search the NE solution in the following.

III. Proposed DDRL Framework

A. DRQN

In the APCC stochastic game, the exploration of each APC is a partial observation Markov decision process (POMDP). In this paper, the DRQN algorithm is proposed to solve the problem with partial observation. Based on the basic structure of DQN, DRQN replaces the first post-convolutional fully-connected layer with a recurrent long short-term memory (LSTM) network [17]. In the training process, the convolutional layer and LSTM layer will be updated together. The Q value obtained by partial observation o_t can be much closer to the real Q value, which is $Q(o_t, a_t; \theta) \rightarrow Q(s_t, a_t; \theta)$.

At time step *t*, after obtaining a state s_t from the environment, the agent will estimate Q-value through a fully connected neural network and choose actions corresponding to the maximum of Q-value. DRQN agents will get a reward r_t and the state s_{t+1} at the next time step. Then, the experience $e(s_t, a_t, r_t, s_{t+1})$ will be stored in a dataset named "replay buffer", which helps DRQN to eliminate the relationship between independent identically distributed training datasets.

Considering that the Q-learning algorithm may overestimate action values under certain conditions, a double Q network is applied to make the Q function fit better. In the process of agent training, a main network Q is primarily used to find out the action a_{t+1} with the maximum Q value at the next time step, and then a target network $\hat{Q}$, which has the same structure as the Q-network, is adopted to estimate the Q-value of the action a_{t+1}. Due to the fact that the target Q-value of the action a_{t+1} may not be the maximum in $\hat{Q}$, this procedure can effectively avoid the overestimation of the suboptimal actions. Noticed that the weights of the main Q-network are copied to the target network at regular time step.

After drawing from the replay buffer, the main Q-network is trained by minimizing a sequence of the loss function

$$L(\theta) = E_{(s_t, a_t) \sim p}\left[\left(y_i - Q(s_t, a_t; \theta, \alpha, \beta)\right)^2\right] \tag{12}$$

where y_i is the target Q-value for iteration *i* computed by target network $\hat{Q}$, and *p* is the probability distribution of state-action pair (s_t, a_t). The updating of DRQN network weights can be done by stochastic gradient with the gradient of the loss function $L(\theta)$. To avoid the algorithm falling into local optimum, a soft update method is adopted, which is

$$\hat{\theta} \leftarrow \zeta\theta + (1 - \zeta)\hat{\theta} \tag{13}$$

where θ and $\hat{\theta}$ are the parameters of the main Q network and the target Q network, respectively. ζ is the coefficient of soft update.

B. QMIX Method

One main issue of DDRL is how to effectively learn the function and fit a proper approximation function, when the parameters of the joint action-value function increase exponentially with the number of agents. Considering the complexity of the environment and the uncertainty of inter-agent communication, the problem of DDRL is actually aimed at a decentralized partially observable Markov decision process (Dec-POMDP) [18].

QMIX is a monotonic value function factorization for DDRL, which can maximize the joint action-value function in a Dec-POMDP. This approach utilizes a hybrid network to combine local value functions of agents and use global states information in the training process, to improve the performance of the DON algorithm. Notice that QMIX adopts the framework of "centralized training and distributed execution", which uses global states in training and partial observations in execution.

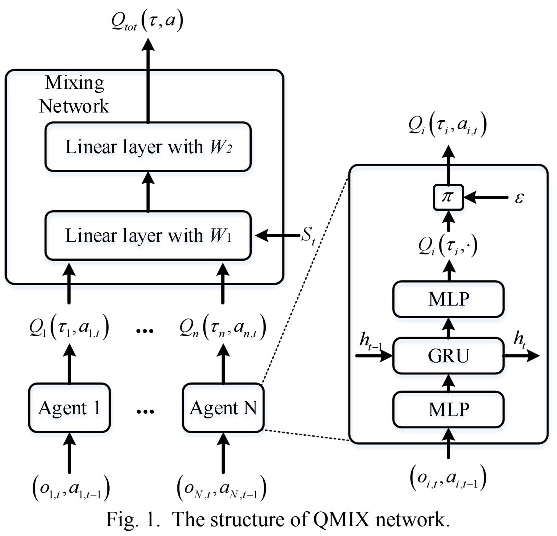


Fig. 1. The structure of QMIX network.

Assumed that, at time slot *t*, agent *i* has partial observation $o_{i,t}$, action $a_{i,t}$, and the global states of the system is s_t. Thus, $\boldsymbol{\tau} = \left[\tau_1,...,\tau_i,...,\tau_n\right]$ is the joint action-observation history, τ_i is the action-observation history of a single agent,

$\mathbf{a}=\left[a_1,...,a_i,...,a_n\right]$ is the joint action of multi-agents. $\pi_i\left(\tau_i\right)$ and $Q_i\left(\tau_i,a_i;\theta_i\right)$ are the strategy and Q function of agent *i*, respectively. $Q_{tot}\left(Q_1,...,Q_n\right)$ is the joint Q function of multi-agents. It can be seen that $Q_i\left(\tau_i,a_i;\theta_i\right)$ is related to τ_i, not to global states information s_t.

The structure of the QMIX network is shown in Fig. 1. The mixing network uses positive weights W_1 and W_2 to satisfy the monotonicity constraint, which is

$$\frac{\partial Q_{tot}}{\partial Q_i}\geq 0,\forall i\in\{1,2,...,n\} \tag{14}$$

As the Q functions of agents have the same monotonicity in the QMIX network, the maximization of the joint Q function is equivalent to the maximization of each local Q function. Therefore, we can obtain the optimal joint strategy by

$$\text{argmax}_{\theta}\,Q_{tot}\left(\boldsymbol{\tau},\mathbf{a};\boldsymbol{\theta}\right)=\begin{pmatrix}\arg\max_{\theta_1}Q_1\left(\tau_1,a_1;\theta_1\right)\\ \cdots\\ \arg\max_{\theta_n}Q_n\left(\tau_n,a_n;\theta_n\right)\end{pmatrix} \tag{16}$$

As shown in Fig. 1, each agent adopts DRQN to fit its Q function $Q_i\left(\tau_i,a_i;\theta_i\right)$. Notice that, considering the inputs of the APCC problem are values, we replace the two convolutional layers with two full-connected layers in DRQN. Besides, we choose GRU rather than LSTM in the Recurrent part since GRU is less computationally expensive. Through a greedy algorithm with parameter ε, DRQN uses the observation $o_{i,t}$ and the action $a_{i,t-1}$ as input to calculate the Q value.

Finally, the loss function of QMIX is presented as

$$L(\theta)=\sum_{i=1}^{n}E_{(\mathbf{s},\mathbf{a})\sim p}\left[\left(y_i^{tot}-Q_{tot}\left(\boldsymbol{\tau},\mathbf{a},\mathbf{s};\boldsymbol{\theta}\right)\right)^2\right] \tag{17}$$

where $y_i^{tot}=r_i+\gamma_i\max_{a'}\hat{Q}\left(\boldsymbol{\tau}',\mathbf{a}',\mathbf{s}';\hat{\boldsymbol{\theta}}\right)$, $\boldsymbol{\theta}=\left[\theta_1,...,\theta_n\right]$.

C. Centralized Training Algorithm of DDRL

Based on the QMIX method introduced in Section III-B, this paper proposed a training structure of DDRL to obtain the NE of the APCC stochastic game. Before starting the training, the networks are initialized with random weights and a replay buffer is established. In each episode, APCs obtain their partial observations from the system state at the beginning. Before APCs take their actions, we perform "do-nothing" actions for each agent in the simulation system, which can predict the next observation provided by the Grid2Op platform [19]. This procedure is to check whether the system is in danger or not. We predefine that the danger status is the line capacity usage of any lines above 0.95 or system failure. If the system is not in danger, APCs will still take "do-nothing" actions. When the danger flag is true in the simulation system, APCs will explore their actions in their action spaces.

Considering that the action space of each APC is large enough, we should guide the exploration of each APC to improve the algorithm performance. Before exploring, each APC will explore all actions if there are any disconnected lines in its control region. Otherwise, it only explore N_k actions in the front rank of all actions' Q-value. For short, these N_k actions are called "top N_k actions". Among the "top N_k actions", the action with best simulation reward will be chosen. Notice that another N_{kb} actions, which is in the front rank exclude the "top N_k actions", will be explored if there are no valid or effective actions. Furthermore, we will choose an action with best reward in historical actions when the APC cannot explore a feasible action.

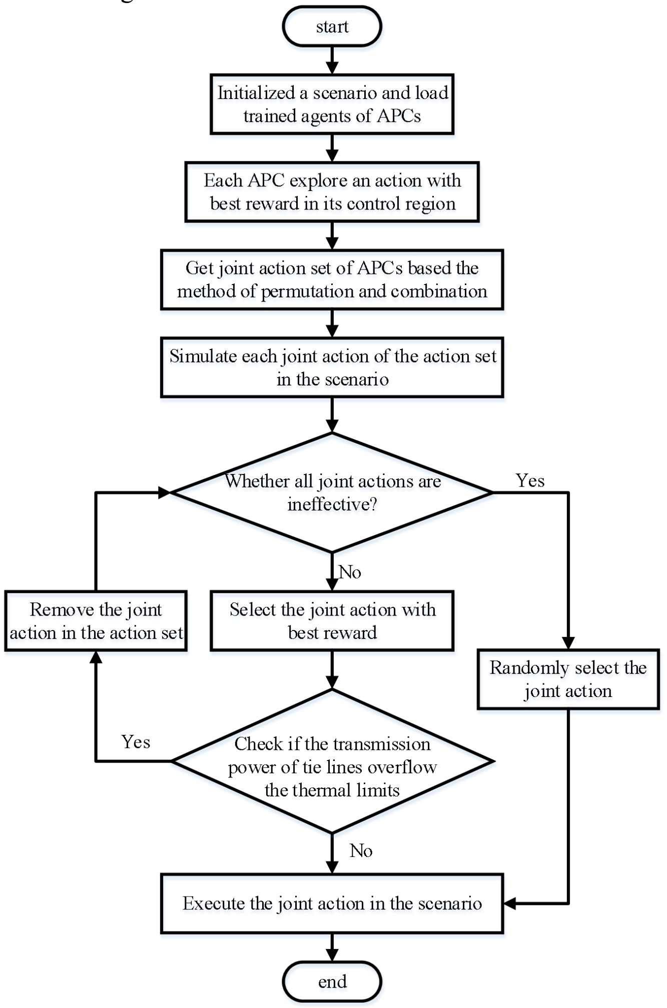
Algorithm 1 Distributed Active Power Correction Control (DAPCC)

```
1:  Initialize DRQN network for each APC with random weights $\theta_i$, initialize mixing network with random weights $W_1$ and $W_2$, initialize Replay Buffer $\Delta$
2:  for episode = 1 to $I$ do
3:    Reset the environment
4:    for t= 1 to $T$ do
5:      Obtain the partial observations $o_{1,t},...,o_{i,t},...,o_{N,t}$ of each agent from state $s_t$, check the danger flag (True for overload or system failure, False for normal state)
6:      if danger flag is True then
7:        for $i$ = 1 to N do
8:          Choose the feasible action with best simulation reward in $N_k$ in the condition of the gameover flag in simulation is True (True for overload or system failure, False for normal state)
9:          if feasible action is None then
10:           Choose the feasible action with best simulation reward in $N_{kb}$ in the condition of the gameover flag in simulation is True
11:           if feasible action is None then
12:             choose an action with best reward in historical actions
13:           end if
14:         end if
15:       end for
16:       Validate the actions in the simulation system and combine each agent's action as the output action $a^t=\left(a_1^t,...,a_i^t,...,a_N^t\right)$
17:     else then
18:       select do-nothing action $\left(a_1^0,...,a_i^0,...,a_N^0\right)$ as the output action $a_t$
19:     end if
20:     Execute action $a^t$ in the environment and obtain next state $s_{t+1}$, reward $r_t$ and $d_t$, store the transition $\left(s_t,a^t,s_{t+1},r_t,d_t\right)_\tau$ in $\Delta$
21:     Sample a batch of transitions from $\Delta$
22:     Calculating the Q-values in target QMIX network:
        $y_i^{tot}=\begin{cases}r_i & \text{if } d_t \text{ is True}\\ r_i+\gamma_i\max_{a'}\hat{Q}\left(\tau',a',s';\hat{\theta}\right) & \text{otherwise}\end{cases}$
23:     Update QMIX-network by losses:
        $L(\theta)=\sum_{i=1}^{n}E_{(s,a)\sim p}\left[\left(y_i^{tot}-Q_{tot}\left(\tau,a,s;\theta\right)\right)^2\right]$
24:     Hard copy main network weights $\theta$ to the target network weights $\hat{\theta}$ regularly
25:     Update state $s_t=s_{t+1}$
26:   end for
27: end for
```

After action selection is completed, the joint action composed of APCs' selections will be executed in the environment. After obtaining the next state and reward of APCs, we store $\left(s_t,a^t,s_{t+1},r_t,d_t\right)_\tau$ to the replay buffer with certain rules. When the memory size of the replay buffer reaches a threshold, batch samples are used to calculate the Q-values in the target QMIX network. The parameters of the QMIX network will update based on the loss computed in (17). The training procedure operates iteratively until a specified number of episodes is reached. The algorithm is denoted as DDRL Training Method for APCC, which is illustrated in Algorithm 1.

D. Distributed Execution Structure for APCC

Based on the training algorithm proposed in Section III-C, we summarize an online execution procedure to obtain the NE of the APCC game. In APCC, the main purpose of agents is to maintain the normal operation of the power grid under different operation conditions. The system characteristics of high dimensionality and non-linearity make it difficult to predict the impact of each control action. For example, when the disturbance is small, the strategy of only a few agents taking actions may outperform the strategy of all the agents taking actions. Hence, an action optimization mechanism is proposed to obtain the optimal joint actions of APCs, whose flowchart is shown in Fig. 2.


start
Initialized a scenario and load trained agents of APCs
Each APC explore an action with best reward in its control region
Get joint action set of APCs based the method of permutation and combination
Simulate each joint action of the action set in the scenario
Whether all joint actions are ineffective?
Yes
No
Remove the joint action in the action set
Select the joint action with best reward
Randomly select the joint action
Yes
Check if the transmission power of tie lines overflow the thermal limits
No
Execute the joint action in the scenario
end
Fig. 2. The flowchart of distributed execution.

We use the method of permutation and combination to obtain the joint action set of APCs. This mechanism can improve the adaptability of multiple agents control. The action exploring each APC is the same as the method mentioned in Algorithm 1. When executing in a scenario, we need to simulate each joint action of the action set to figure out whether it is effective or not. If all actions in the action set are ineffective, a randomly selected joint action will be adopted. Otherwise, we will select the joint action with the best reward. After action selection, the transmission power of tie lines will be check to determine whether the joint action will be executed or not. If the joint action will lead to overload of any tie lines, we will remove it and search for another feasible joint action. It can be seen that we primarily consider the transmission power of tie lines to ensure the stability of power flow interfaces.

IV. Case Study

In this section, the proposed DAPC method is evaluated on an open-source platform "Grid2Op". To provide a typical application scenario for the DAPCC application, the 118-bus grid provided by "2020 Learning to Run a Power Network - Neurips Track 2" global competition cases [20] are used to evaluate the algorithm performance. In the 118-bus grid, we divided the centralized control region into three distributed control regions, which is shown in Fig. 3. Three agents based on the QMIX method are implemented to control the elements of three regions, respectively. We choose a multi-mix dataset, that is a set of cases with varying amount of renewables shown in Fig. 4.

Each mixed dataset includes 576 scenarios covering each month in 48 years. Each scenario contains data for 28 continuous days with 5-minute resolution. The penetration rate of RESs will change in different scenarios, and the maintenance of lines is considered. We need to train our agents to adapt to renewable energy productions in the grid with an increasing rate of less controllable renewable energies over years. After training, the trained agents will be tested on hidden new mixed dataset that are not present in the training set, to assess the adaptability of your agent. The 24 test scenarios are randomly extract from every months, which cover the characteristics of typical scenarios through one year.

As we implement three agents in our case study, the network structure of the QMIX method consists of three identical DRQNs and a mixing network. The DRQNs are composed of two full-connected layers and a GRU layer with 512 neurons. The mixing network is composed of DNN with 256 neurons. The rectified linear unit (ReLU) is used as an activation function, and the batch size is set to 64. RMS is adopted for the QMIX network and the learning rate is set as 0.0005. The maximum number of steps in an episode is set to 4000. The reaction time and recovery time of the APCC problem are set to 3 time steps, i.e., 15 minutes. The simulations are conducted on a Linux server with 3 GPUs.

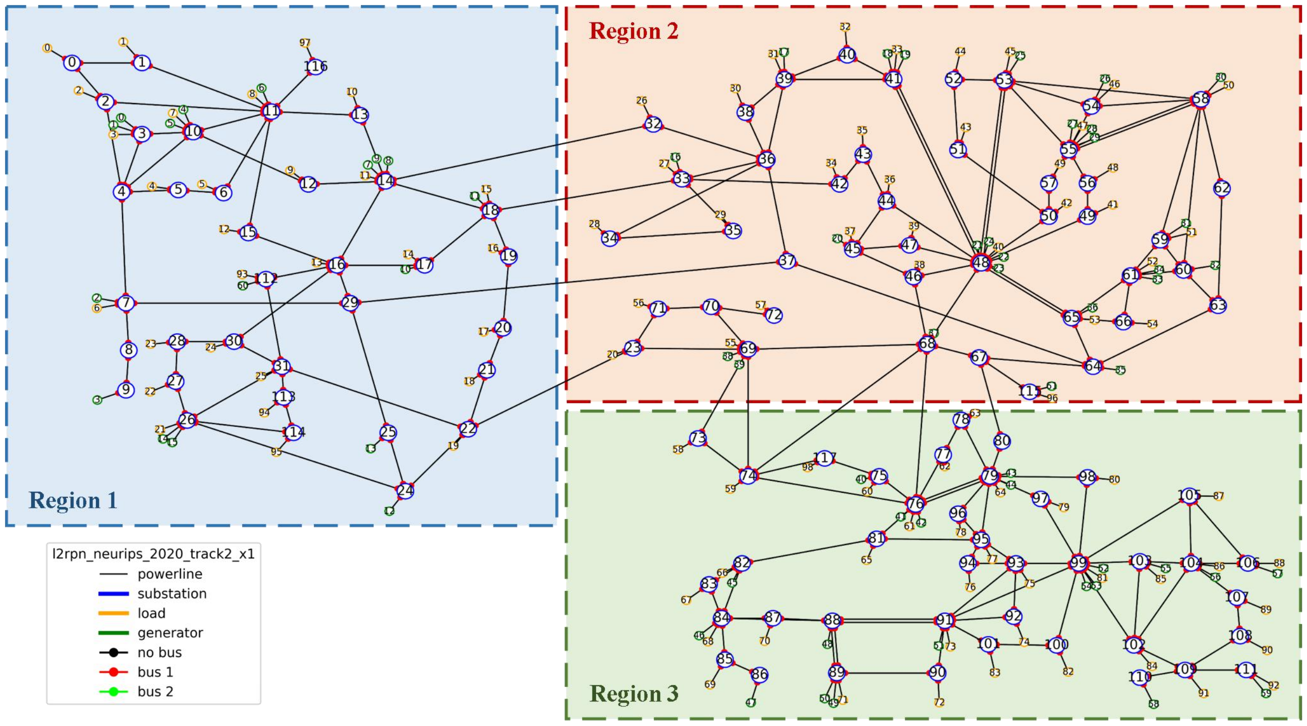


Fig. 3. The topology of the 118-bus power grid

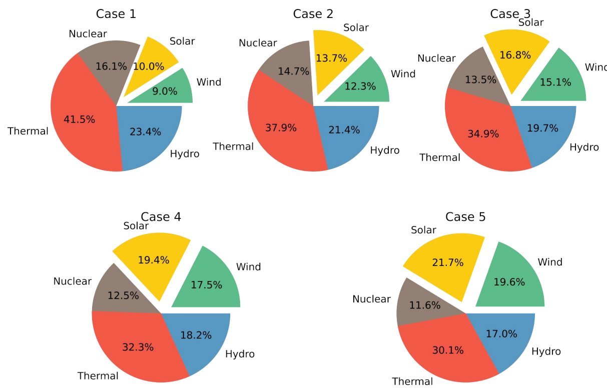


Fig. 4. Energy profiles of each case.

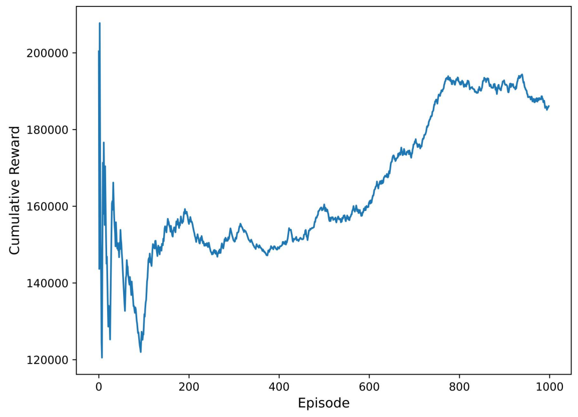


Fig. 5. The moving average curve of the cumulative reward of proposed DAPCC.

A. Centralized Learning Capability

In this part, we adopted the online centralized training algorithm proposed in Section III-C to train the three APC agents. Through the Monte Carlo tree search (MCTS) described in Part I, the amounts of selected effective actions of Region 1, Region 2, and Region 3 are 46, 540, and 565, respectively. It can be seen from the results of MCTS that the network complexity of Region 2 and Region 3 is higher than that of Region 1. The effective action matrixes of three regions are used as action spaces for three APCs, respectively. Specifically, we add "do-nothing" action to the action space of each APC, which is explained in Section III-C. In each episode, the scenarios of various RES penetration rate contained in every cases are randomly selected in the training process.

After 1000 episodes of training, we take the moving average value of cumulative rewards for every 200 episodes, the cumulative reward curve of DAPCC is shown in Fig. 5. In the early phase of training, the scenarios with a variable penetration rate of RESs are randomly selected, as a result, the trend of cumulative reward varies dramatically. From 200 episodes to 600 episodes, the curve of cumulative reward becomes stable and increasing. It can be seen that the cumulative reward can finally begin to converge at around 800 episodes, which indicates the NE of the APCC game might be obtained.

B. Distributed Executing Performance

In this part, we implement the trained APC agents in Section IV-A to evaluate whether they can solve the problem of APCC. The information of 24 test scenarios is provided in Table 1, which contains all kinds of cases.

In the evaluation process, we test our trained APCs in a sequence of the penetration rate of RESs and limit the maximum alive steps of a scenario to 4000. In Fig. 6, the legend “max-3-actions” (M3A) denotes that the number of APCs' actions allowed in a single time slot is limited to 3, meanwhile, the legend “max-1-actions” (M1A) denotes to be limited to 1. Noticed that the performance of “do-nothing” action is used to perform comparison with other schemes.

TABLE I
THE INFORMATION OF 24 TEST SCENARIOS

Mix ID	Scenario numbers	The active load consumption (MV) in each scenario					
Case 1	3	2163	2217	2147			
Case 2	4	2081	2099	2217	2163		
Case 3	6	2104	2033	2228	2157	2145	2183
Case 4	6	2143	2161	2142	2130	2188	2143
Case 5	5	2093	1999	2175	2156	2110	

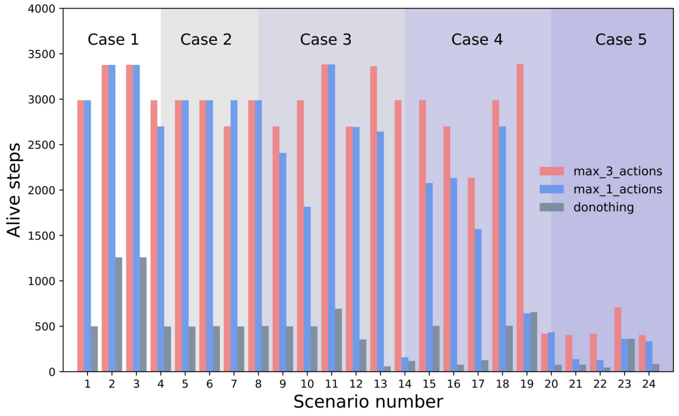


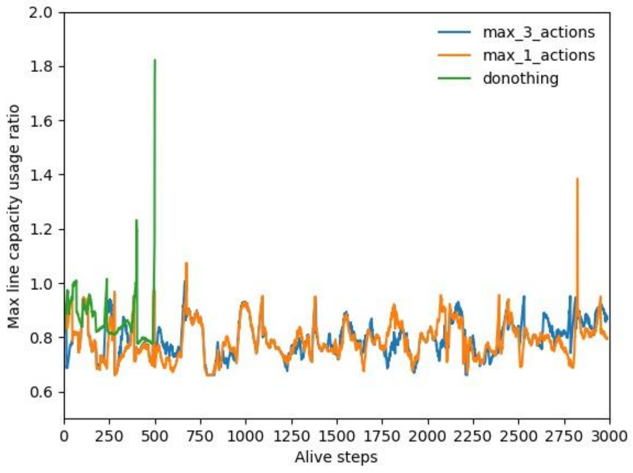
Fig. 6. The performance of DAPCC in 24 test scenarios.

In the first 7 scenarios, the mixed data of environments are case 1 and case 2. It can be seen that the control action of DAPCC is effective, which can make the power system alive longer. Considering the scenarios in case 1 and case 2 contain only 19% RESs, the performance of the M3A scheme is similar to that of the M1A scheme. In the next 12 scenarios of case 3 and case 4, the M3A scheme performs more effectively and stably than the M1A scheme, which indicates the distributed executing scheme we proposed is beneficial to the APCC problem. The last 5 scenarios, which contain 41.3% RESs in case 5, are difficult for our agent with only BBS control. However, the APCs with the M3A scheme is also better than other action schemes. Notice that the average decision time for each time step of case 1, case 2 , case 3 and case 4 is around 40 milliseconds, this indicate that the proposed method has practicability in APCC problem.

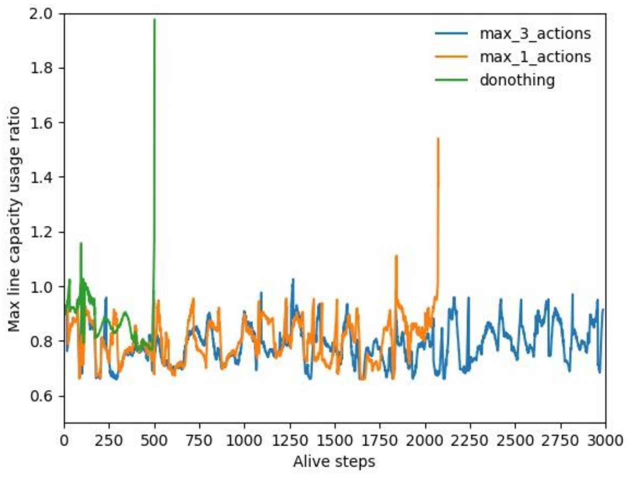
C. Adaptability of proposed control method

To fully evaluate the performance of DAPCC, we randomly selected two scenarios, which are in case 2 and case 4, to test our trained APCs. The comparison of M3A action scheme and M1A action scheme is still set up in this part. The alive steps of the M3A scheme and M1A scheme are almost the same in the scenario of case 2, while they are different in the scenario of case 4.

We firstly investigate the maximum line capacity usage of each control region to directly reflect the effectiveness of the control strategy. As shown in Fig. 7, the system will blackout with a “do-nothing” action at around 500 steps. This illustrates that the plugged-in RESs will lead to heavy overload in the power system if no control measure is taken. In Fig. 7(a), although the time steps during which the system is “alive” using the M1A scheme and M3A scheme both reach 3000 steps, the maximum line capacity usage ratio of the M1A scheme increase sharply at around 2800 steps, which indicate that the robustness of the M3A scheme is better than that of M1A scheme. From Fig. 7(b), it can be seen that, with the penetration rate of RESs increasing, the “do-nothing” scheme and M1A scheme cannot eliminate or alleviate the overload as the M3A scheme do. This demonstrates the adaptability of our proposed control method.

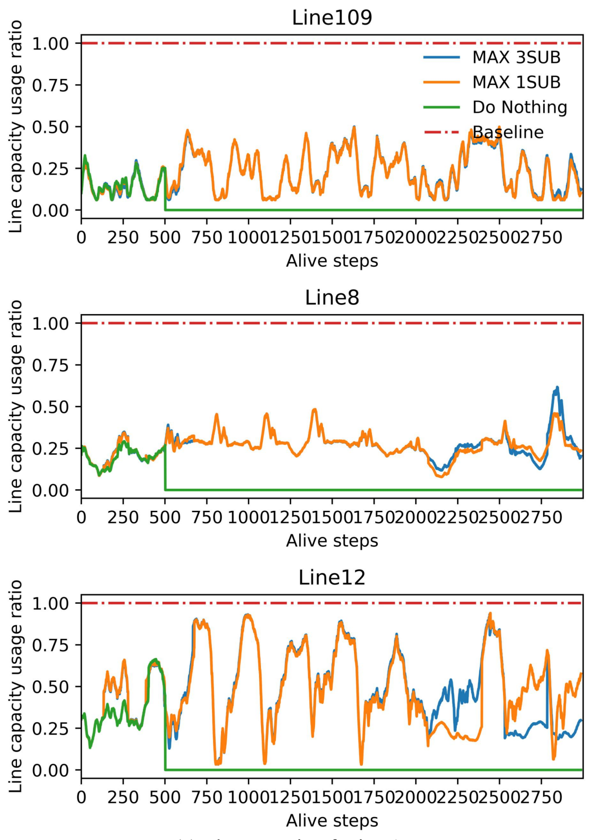


(a) System operating status under different schemes in the scenario of case 2.

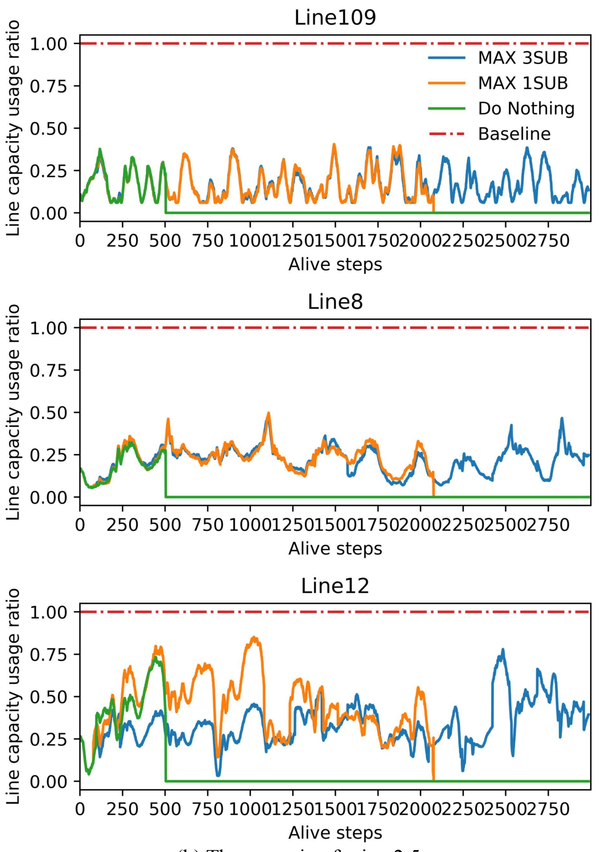


(b) System operating status under different schemes in the scenario of case 4.
Fig. 7. The comparison of system operating status under different schemes.

After investigating the control performance of each APC, we now focus on the line capacity usage of tie lines to evaluate the coordination between the APCs. The parameters of tie lines are shown in Table II. We record the line capacity usage in the test scenarios of case 2 and case 4. As shown in Fig. 3, the bus 18, 33, 68, 69, 74, and 76, which connected more loads and lines than others, are regarded as heavy load buses. Hence, we check the transmission power of the line 109, line 8, and line 12, which is shown in Fig. 8. It can be seen that the line capacity usage of these three tie lines can always be controlled below the baseline. This illustrates that the DAPCC method is available for multi-region coordinative control.



(a) The scenario of mix x1.5



(b) The scenario of mix x2.5

Fig. 8. The line capacity usage of tie lines in the scenarios

TABLE II
THE PARAMETERS OF TIE LINES

Line ID	Line Origin bus	Line extremity bus	Transmission capacity (MW)
108	14	32	208.10
109	18	33	**226.90**
117	29	37	456.00
94	22	23	674.20
7	69	73	480.00
8	69	74	**436.50**
9	68	74	681.80
12	68	76	**682.50**
185	67	80	997.10

V. CONCLUSIONS

To improve the adaptability of the DRL algorithm applied in the APCC problem, a DAPCC method is proposed to conduct the framework of "centralized training and distributed executing" scheme for a large-scale power system. Based on the theory of the stochastic game, we adopted a QMIX method to obtain the NE of APCs. Considering cooperation of APCs, an action-optimization mechanism is proposed to obtain the global optimality of joint actions. The case studies demonstrate the convergence of centralized learning capability and the adaptability of distributed executing performance in large-scale APCC with plugged-in RESs. The application of DRL for controlling complex and complicated power system is still in its infant form, the future work on this topic will be with multi-faceted application scenarios, and expected to solve power system control problems with ultra-high dimensionality and nonlinearity.

REFERENCES

[1] X. Li, P. Balasubramanian, M. Sahraei-Ardakani, M. Abdi-Khorsand, K. W. Hedman and R. Podmore, "Real-Time Contingency Analysis With Corrective Transmission Switching," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 2604-2617, July 2017.

[2] H. Diao, M. Yang, F. Chen and G. Sun, "Reactive power and voltage optimization control approach of the regional power grid based on reinforcement learning theory," *Transactions of China Electrotechnical Society*, vol. 30, no. 12, pp. 408-414, Dec. 2015.

[3] J. Zhang, C. Liu, S. Jennie, J. Song and Y. Su, "Deep reinforcement leaming for short-term voltage control by dynamic load shedding in china southern power grid". In *IJCNN*, 2018, pp. 1-8.

[4] Q. Yang, G. Wang, A. Sadeghi, G. B. Giannakis and J. Sun, "Two-timescale voltage control in distribution grids using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2313-2323, May 2020.

[5] J. Duan, D. Shi, R. Diao, H. Li, Z. Wang, B. Zhang, D. Bian and Z. Yi, "Deep-Reinforcement-Learning-Based Autonomous Voltage Control for Power Grid Operations," *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 814-817, Jan. 2020.

[6] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, et al, "Starcraft ii: A new challenge for reinforcement learning," *arXiv e-prints*, Aug. 2017

[7] B. K. Petersen, J. Yang, W. S. Grathwohl, C. Cockrell, C. Santiago, G. An and D. M. Faissol, "Deep reinforcement learning and simulation as a path toward precision medicine," *Journal of Computational Biology*, vol. 26, no. 6, pp. 597-604, Jun. 2019.

[8] J. Chen, B. Yuan, M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," In *IEEE ITSC*, 2019, pp. 2765-2771.

[9] C. H. Liu, Z. Chen, J. Tang, J. Xu and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep

reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059-2070, Sept. 2018.

[10] I. Adamski, R. Adamski, T. Grel, A. Jędrych, K. Kaczmarek and H. Michalewski, "Distributed deep reinforcement learning: Learn how to play atari games in 21 minutes," In *Int. Conf. High Performance Computing*, 2018, pp. 370-388.

[11] K. Lin, R. Zhao, Z. Xu and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," In *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 2018, pp. 1774-1783.

[12] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th int. conf. machine learning*, 1993, pp. 330–337.

[13] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls and T. Graepel, "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward," In *AAMAS*, 2018, pp. 2085-2087.

[14] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv e-prints*, Mar. 2018.

[15] A. Marot, B. Donnot, C. Romero, B. Donon, M. Lerousseau, L. Veyrin-Forrer and I. Guyon, "Learning to run a power network challenge for training topology controllers," *Electric Power Systems Research*, vol. *189*, no. 106635, Dec. 2020.

[16] Universidad Nacional de Colombia, L2RPN-NEURIPS-2020 [Online]. Available: https://github.com/unaioperator/l2rpn-neurips-2020.

[17] M. Hausknecht, P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv e-prints*, Jul. 2015.

[18] C. Amato, G. Chowdhary, A. Geramifard, N. K. Üre and M. J. Kochenderfer, "Decentralized control of partially observable Markov decision processes," In *52nd IEEE Conf. Decision and Control*, pp. 2398-2405, 2013.

[19] RTE-France, Grid2Op [Online]. Available: https://github.com/rte-france/Grid2Op.

[20] RTE-France, L2RPN NEURIPS 2020 - Robustness Track [Online]. Available: https://competitions.codalab.org/competitions/25426.