Energy Reports 7 (2021) 8900–8911

Contents lists available at ScienceDirect

Energy Reports

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

Research paper

Blockchain Enabled Intelligence of Federated Systems (BELIEFS): An attack-tolerant trustable distributed intelligence paradigm

Siyuan Chen [a], Jun Zhang [a,*], Yuyang Bai [a], Peidong Xu [a], Tianlu Gao [a], Huaiguang Jiang [b], Wenzhong Gao [c], Xiang Li [a]

[a] School of Electrical Engineering and Automation, Wuhan University, Wuhan, 430072, Hubei Province, China
[b] National Renewable Energy Laboratory, Golden, Denver, 80401, CO, USA
[c] Department of Electrical and Computer Engineering, University of Denver, Denver, 80208, CO, USA

ARTICLE INFO

Article history:
Received 2 September 2021
Received in revised form 26 October 2021
Accepted 31 October 2021
Available online 3 December 2021

Keywords:
Multi-regional large-scale power system
Multi-agents system
Blockchain enabled intelligence
Attack-tolerant capability
Distributed deep reinforcement learning

ABSTRACT

In this article, a Blockchain Enabled Intelligence of Federated Systems (BELIEFS) is proposed to conduct cooperative control for the multi-regional large-scale power system with a multi-agents system (MAS). By establishing a two levels blockchain, each regional AI agent can simultaneously manage intra-regional controllers and cooperate with other AI agents. Under the consensus mechanism, the agents, which respectively conducted distributed deep reinforcement learning (DDRL) algorithm in multi-regions, can have the tolerant capability of malicious attacks in their training process. The demonstration of the proposed approach is within a multi-regional large-scale interconnected power system. Under the mode of "centralized dispatching and hierarchical management", this article aims to definite a mathematical model to deal with the control problem of the power systems. With the comparison experiments, the effectiveness and efficiency of our proposed method in the training process are verified. In addition, malicious attacks are set on the main chain and shard chains to verify the attack-tolerant capability. We expect that such approach and results can suggest a new paradigm of attack-tolerant trustable distributed AI deployment.

1. Introduction

In recent years, with the rapidly growing scale, increasingly interconnectivity and complexity of the power grid, a multi-regional, large-scale power system (MLPS) is divided into multiple operating sub-regions according to the power flow sections. Generally, a MLPS adopts a multi-agent system (MAS), consisting of a group of cooperative autonomous agents with communication, to meet certain coordinated task requirements. Current methods require amounts of data interaction and calculation to solve the problems of the large-scale inter-connected systems and hardly have attack-tolerant capability for their performance relying on the specific structural topology and the data authenticity. Thus, the traditional methods will face several challenges: (1) the attack-tolerant capability of the conventional method is insufficient; (2) In data interaction process, the security and privacy of operation data in each region cannot be guaranteed; (3) the pattern of centralized control methods has low efficiency and great difficulty for resolving the problem. Therefore, an attack-tolerant, trustable and distributed intelligent paradigm is essential to tackle out the significant tasks of the MLPS.

As a fault caused by attacks in one agent may spreads throughout the MAS via the communication network, the control issues, especially for the MLPSs, have strict demand of the reliability and resilience of each system component or agent. Attack tolerant control (ATC) is designed against the attacks or failures of a MLPS which may lead to degradation of system performance or even cause instability (Chen et al., 2021). Jin et al. (2021) developed distributed security control strategies to achieve the bounded consensus of the multi-agent systems under the influence of network decay and intermittent attacks. Consensus algorithm can mathematically solve the problem with local or neighboring information through a consensus mechanism, which is a potential technique to conduct attack tolerant control scheme (Chen et al., 2013; Qin et al., 2017). Hug et al. (2015) proposed a fully distributed control method to solve the optimal power flow (OPF) problem based on the consensus + innovations method. The convergence analysis proves that the control strategy will lead to a unique optimal solution. However, these consensus-based methods, which mostly need several iterations to achieve the convergence, cannot solve the high-dimensional and non-convex problem quickly and efficiently.

* Corresponding author.
E-mail address: jun.zhang.ee@whu.edu.cn (J. Zhang).

https://doi.org/10.1016/j.egyr.2021.10.113

Nomenclatures and Abbreviation

α_k	The innovation weight at kth iteration
β_k	The consensus weight at kth iteration
$\boldsymbol{\Omega}_i$	The neighbor set of region i
$\boldsymbol{\pi}^*$	The optimal joint strategy vector
$\boldsymbol{\tau}_{0-k}$	The trajectory from state s_0 to state s_k
$\boldsymbol{\vartheta}$	The set of fault caused by malicious attacks
$\boldsymbol{A}_i$	The partial observation space of agent i
$\boldsymbol{N}_{ag}$	The set of the agents
$\boldsymbol{O}_i$	The partial observation space of agent i
$\boldsymbol{S}$	The state space of the MLPS
$\boldsymbol{u}$	The set of control strategies
$\boldsymbol{x}$	The set of system states
γ_i	The discount factor of agent i
$\hat{\theta}$	The parameters of the target Q network
π_i	The strategy of agent i
π_i^*	The optimal strategy of agent i
π_i^+	The arbitrary strategy of agent i
τ_{s_0}	The initial state of the trajectory
θ	The parameters of the main Q network
ϑ_i^c	Cooperation fault
ϑ_i^d	Data collection fault
ϑ_i^e	Agent execution fault
ϑ_i^n	Network communication fault
ζ	The coefficient of soft update
$a_{i,t}$	The control strategy of agent i at time t
$A_i(\bullet)$	The advantage function of agent i
$F(\bullet)$	The objective function
$f_i(\bullet)$	The vector field of region i
$G(\bullet)$	The equation constraints
$g_i(\bullet)$	The control vector field of region i
$H(\bullet)$	The equation constraints
$L(\bullet)$	The loss function
N_{ag}	The number of the agents
N_p	The sampling number in Q-learning algorithm
$o_{i,t}$	The observation of agent i at time t
p	The state transition probability
p_1, p_2, τ_1, τ_2	The hyper-parameters of QD-learning
$Q_i(\bullet)$	The state–action function of agent i
R_c	The positive constant corresponding to the total generation cost
R_{div}	The negative constant corresponding to the divergence of power flow
$r_{i,t}$	The immediate reward of agent i at time t
$R_i(\bullet)$	The immediate reward of agent i
R_{rho}	The negative coefficients corresponding to the heavy load case
R_r	The negative coefficients corresponding to the total generation cost
s_t	The state of MLPS at time t
u_i	The control strategy of region i
$V_i(\bullet)$	The value function of agent i
x_i	The system state of region i
ACOPF	Alternating current optimal power flow
AI	Artificial intelligence
ATC	Attack tolerant control
DDPG	Deep deterministic policy gradient
DDRL	Distributed deep reinforcement learning
Dec-POMDP	Decentralized partially observable Markov decision process
DQN	Deep Q network
DRL	Deep reinforcement learning
GLP	Graph Laplacian potential
IPM	Interior Point Method
IQL	Independent Q learning
MAS	Multi-agents system
MESDDDPG	Multi-role exploration strategy distributed deep deterministic policy gradient
MLPS	Multi-regional, large-scale power
MNE	Mixed-strategy Nash equilibrium
OPF	Optimal power flow
RELU	Rectified linear unit
RL	Reinforcement learning

The ATC, however, is difficult to obtain the data of the whole MLPS in a MLPS because the multiple sub-regions generally operate in the mode of federated systems. Therefore, a decentralized trustable information and communication technology is essential to ensure the security and privacy of the data belonged to different sub-regions. Blockchain is an emerging technology for distributed information and computing infrastructure, which provide a platform to eliminate the crisis of trust in the data interaction of a MLPS. Authors in Ahmed et al. (2021) applied a peer-to-peer blockchain in a interconnected power system and described the simulation analysis and reduction of harmonics detection in a peer-to-peer interconnected 3-phase power system. Authors in Foti et al. (2021) proposed a distributed multi-area OPF algorithm based on blockchain consensus mechanism, which can restrain the improper profit-making behavior of malicious attacks and ensure the impartiality of distributed optimization operation. In the paper (Haoran et al., 2015), the writers design, implement and analyze a decentralized consensus algorithm based on blockchain technology to solve the OPF problem. The proposed algorithm enables independent nodes in power grid, to reach an agreement on the optimal power flow solution without prior trust in each other.

Deep reinforcement learning (DRL) is a combination of deep learning (DL) and reinforcement learning (RL)technology, which can directly learn from agents' interactions with an environment (Zhang et al., 2018b; Yang et al., 2020; Su et al., 2021). In recent years, DRl has been widely applied in the field of power system (Li et al., 2021; Liu et al., 2020). Paper (Sanaye and Sarrafi, 2021) proposed an energy management method uses the principles of Reinforcement Learning (RL) based on Deep Q Network (DQN) algorithm to find the operational strategy for the hybrid system. As an extension technology of DRL, distributed deep reinforcement learning (DDRL) has been successfully applied in multi-agent systems (MAS) manner and effectively solved the problem of high-dimensional action space faced by DRL (Lin et al., 2018). In Tan (1993), combining independent Q-learning (IQL) with DQN, a DDRL framework is proposed to make each agent has an independent Q network. However, the environment for each agent is unknown and non-static because the IQL algorithm does not conduct interactions among multiple agents.

The IQL algorithm violates the Markov decision process (MDP) and cannot be proven to achieve the convergence of the algorithm. Recently, DRL makes great success in many control areas such as games (Sunehag et al., 2017; Rashid et al., 2018), robotics (Gu et al., 2017; Duan et al., 2016), autonomous driving (Kiran et al., 2021) and provides a promising optimization algorithm for power system control and operation (Huang et al., 2020; Yan and Xu, 2020a; Duan et al., 2020). Paper (Diao et al., 2019) adopts DRL to derive fast OPF solutions, which can greatly assist power grid operators in making rapid and effective decisions. Paper (Zhou et al., 2020) proposed a real-time optimal power flow approach using Lagrangian-based DRL in the continuous action domain. Paper (Li and Yu, 2021) proposed a novel algorithm for tuning the controller, and termed the multi-role exploration strategy distributed deep deterministic policy gradient (MESDDDPG), which builds upon the deep deterministic policy gradient (DDPG) by employing various trick including a multi-role exploration strategy.

Therefore, an attack-tolerant, trustable and distributed method, which integrated ATC, blockchain framework and DDRL algorithm, is proposed in this paper to address the challenges of the MLPS control problem. Basically, a MAS is implemented for the regional operators to conduct a hierarchy control for the MLPS. Then, a two-levels blockchain is established to provide a trustable platform for the data interaction between the agents of the federated systems. Each agent of the MAS communicates with other agents on the main chain and interact with its intra-regional neighbor nodes of the shard chains. Through the trustful environment of blockchain technology, we can prevent from the data falsification in the interaction between the topology nodes of the MLPS. On the infrastructure, a distributed deep Q learning algorithm, namely QD-learning, is applied to allow agents for their consensus and innovation in a training phase. The consensus mechanism, which conducted in the training process of the QD-learning algorithm can efficiently improve the attack-tolerant capability of MAS. Our proposed approach is named as Blockchain Enabled Intelligence of Federated Systems (BELIEFS), which indicates that we focus on establishing a trustable paradigm to train multiple reliable DDRL-based agents.

This paper provides a basic paradigm for establishing a trusted power grid operation control and dispatch platform. Our proposed method is not used to instead of the existing DRL method. It is a paradigm which can compatibility with other DRL algorithm. Our proposed method aims to avoid the data fault or agent fault caused by the malicious attacks, which support the efficiently training and valid execution of DRL agents. The main contributions of this paper are the following threefold:

(1) a two-levels blockchain structure is applied in the MLPS, which include a main chain with PV-buses as the nodes and several shard chains with PQ-buses as nodes. It is applicable for multiple agents to obtain the reliable and valid data;
(2) a distributed DRL-based intelligence paradigm is established, which can cooperatively deal with the operation control of the AC-OPF problem of the MLPS.
(3) a QD-learning algorithm is applied in the proposed distributed intelligence to improve the learning efficiency and execution reliability of agents, which can have the attack-tolerant capability in case of suffering the malicious attacks.

The remainder of our paper is organized as follows: The problem formulation, framework, and algorithm are described in Section 2. Our proposed framework for the attack-tolerant trustful distributed intelligence paradigm is described in Section 3. The learning capability, execution performance and attack-tolerant capability of the proposed intelligence paradigm is verified by case studies of an interconnected power system in Section 4. The conclusions and suggestions for future work are given in Section 5.

2. Problem formulation, preliminaries and algorithm

2.1. Problem formulation of attack-tolerant tertiary control on MLPS

The tertiary control of MLPS can adopted our proposed attack-tolerant distributed intelligence paradigm. Tertiary control facilitates economic dispatch by setting the power output and voltage set-point of generators, which actually aim to quickly solve an alternating current optimal power flow (ACOPF) problem. Thus, considering the faults caused by attacks, the tertiary control of MLPS is established as the general form of

$$\begin{cases} \dot{x}_i = f_i\left(x_i, \vartheta_i^d\right) + g_i\left(x_i, \vartheta_i^e\right) u_i\left(x_i, x_j, \vartheta_i^n\right), j \in \Omega_i \\ y = F\left(x, \vartheta_i^c\right) \\ H(x) = 0 \\ G(x) \le 0 \end{cases} \tag{1}$$

where $x = \{x_1, x_2, \ldots, x_n\}$ is the set of system states with x_i being the state of region i, and Ω_i denotes the neighbor set of region i.$u = \{u_1, u_2, \ldots, u_n\}$ is the set of control strategies with u_i being the control strategy of region i include active generation, reactive generation, voltage magnitude and bus angle.$f_i(\bullet)$ and $g_i(\bullet)$ are vector field and control vector field of region i, respectively. ϑ_i^d, ϑ_i^e, ϑ_i^n and ϑ_i^c represent data collection fault, agent execution fault, network communication fault and cooperation fault caused by malicious attacks, respectively. $F_i(\bullet)$, $H_i(\bullet)$ and $G_i(\bullet)$ are objective functions, equation constraints and inequation constraints.y is the states of the MLPS under the objective function. Assumed that $\vartheta = \{\vartheta_1, \vartheta_2, \ldots, \vartheta_n\}$ is the set of faults, which caused by malicious attacks occurring in MLPS, with $\vartheta_i = \left\{\vartheta_i^d, \vartheta_i^e, \vartheta_i^n, \vartheta_i^c\right\}$ being the fault of region i.

2.2. Stochastic game and mixed-strategy Nash equilibrium

A multi-agents system is adopted in this paper to deal with "the curse of dimensionality", which can implement a fully cooperative scheme in large-scale power grids. In the framework of DDRL, each agent follows the basic learning paradigm of reinforcement learning and make strategy in the decentralized partially observable Markov decision process (Dec-POMDP). The Dec-POMDP of multiple agents can be captured in the form of a cooperative stochastic game, whose main components of the game include:

(1) **Agent**: agent i in multiple regions. $i \in \mathbf{N}_{ag}$, where $\mathbf{N}_{ag}$ is the set of the agents;
(2) **State**: the states s_t of the MLPS 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;
(3) **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;
(4) **Action**: the control strategy $a_{i,t}$. $a_{i,t} \in \mathbf{A}_i$, where $\mathbf{A}_i$ is the action space of agent i;
(5) **Reward**: immediate reward $r_{i,t}$.

Each agent will obtain an immediate reward when the power system maintains its normal operation and get much larger negative rewards if power system blackout happens. Thus, the agents perform a fully cooperative stochastic game to achieve a mixed-strategy Nash equilibrium (MNE) based on their observations. The immediate reward of agent i at time k can be formulated as:

$$R\left(s_k\right) = \sum_{t=0}^{k-1} \gamma_i^k r_{t+1} \tag{2}$$

where κ is a negative constant. γ_i is the discount factor of agent i.

To evaluate the long-term return of the agents' strategies, a value function, the expectation of the long-term expected return, is introduced in the Dec-POMDP as follows:

$$\begin{aligned} V_i^{\pi}(s_k) &= E_{\tau_{0-k}\sim p(\tau)}\left[R(s_k)|\,\tau_{s_0}=s_k\right] \\ &= E_{\tau_{0-k}\sim p(\tau)}\left[r_1+\gamma_i\sum_{t=0}^{k-1}\gamma_i{}^k r_{t+1}\middle|\,\tau_{s_0}=s_k\right] \\ &= E_{a_k\sim\pi_i(a_k|s_k)}E_{s_{k+1}\sim p(s_{k+1}|s_k,a_k)}E_{\tau_{1-k}\sim p(\tau)} \\ &\quad\left[r_1+\gamma_i\sum_{t=0}^{k-1}\gamma_i{}^k r_{t+1}\middle|\,\tau_{s_0}=s_{k+1}\right] \\ &= E_{a_k\sim\pi_i(a_k|s_k)}E_{s_{k+1}\sim p(s_{k+1}|s_k,a_k)} \\ &\quad\left[r(s_k,a_k,s_{k+1})+\gamma_i E_{\tau_{1-k}\sim p(\tau)}\right. \\ &\quad\left.\left[\sum_{t=0}^{k-1}\gamma_i{}^k r_{t+1}\middle|\,\tau_{s_0}=s_{k+1}\right]\right] \\ &= E_{a_k\sim\pi_i(a_k|s_k)}E_{s_{k+1}\sim p(s_{k+1}|s_k,a_k)} \\ &\quad\times\left[r(s_k,a_k,s_{k+1})+\gamma_i V_i^{\pi}(s_{k+1})\right] \end{aligned} \tag{3}$$

where τ_{0-k} is a trajectory of $(s_0,a_0,s_1,\ldots,s_k)$, thus $\tau_{0-k}=(s_0,a_0,\tau_{1-k})$. τ_{s_0} is the initial state of the trajectory. $\pi_i(a_k|s_k)$ is the strategy of agent i. $p(s_{k+1}|s_k,a_k)$ is the state transition probability.

The value function of each agent is related to its strategy such that the term $V_i^{\pi}(s_k)$ can be denoted by $V_i(s_k,\pi_i)$. The overall objective of the Dec-POMDP is to achieve agents' deterministic strategies which can maximize their value functions. Therefore, the solution of the stochastic game is MNE, which is a state of the game where no agents can benefit by unilaterally adjusting strategies. Assumed that the MNE solution of ATC problem denotes by $\pi^* = \left[\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_{N_{ag}}^*\right]^T$, the optimality of the MNE solution can be described as:

$$V_i\left(s_k,\pi_i^*\right)\geq V_i\left(s_k,\pi_i^+\right),\forall i\in N_{ag} \tag{4}$$

where π^* is the optimal joint strategy while π_i^* is the optimal strategy of agent i. π_i^+ is arbitrary strategy of agent i besides π_i^*. N_{ag} is the number of the agents.

Based on Eq. (4), the state–action value function can be presented as:

$$Q_i^{\pi}(s_k,a_k)=E_{s_{k+1}\sim p(s_{k+1}|s_k,a_k)}\left[r(s_k,a_k,s_{k+1})+\gamma_i V_i^{\pi}(s_{k+1})\right] \tag{5}$$

where follows that the value function $V_i^{\pi}(s_k)$ is the expectation of $Q_i^{\pi}(s_k,a_k)$ with respect to a_k. According to Eq. (5), the Bellman equation of $Q_i^{\pi}(s_k,a_k)$ can be described as:

$$\begin{aligned} Q_i^{\pi}(s_k,a_k) &= E_{s_{k+1}\sim p(s_{k+1}|s_k,a_k)} \\ &\quad\left[r(s_k,a_k,s_{k+1})+\gamma_i E_{a_{k+1}\sim\pi_i(a_{k+1}|s_{k+1})}\right. \\ &\quad\left.\left[Q_i^{\pi}(s_{k+1},a_{k+1})\right]\right] \end{aligned} \tag{6}$$

Therefore, we can search the optimal strategies of the agents through maximizing the value of $Q_i^{\pi}(s_k,a_k)$. Considering that the $Q_i^{\pi}(s_k,a_k)$ of the agents has the same monotonicity in the fully cooperative task, the MNE solution can be obtained after all agents get the maximum of their state–action value function. To address this issue, in previous literature some model-based methods attempt to obtain the MNE of stochastic games, 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.

2.3. QD-learning algorithm

Graph Laplacian potential (GLP), as a measure of the total disagreement among all agents, is adopted in consensus algorithm to improve the cooperative ability of intelligent agents. The general form of consensus potential is described as follows:

$$C_v^i=\beta_i\sum_{j\in\Omega}\left(v_j-v_i\right) \tag{7}$$

where v_i are the variables which need to reach a consensus. Ω_i is the neighbors set of the node i. β_i is the weight parameter. The iteration calculation achieves the consensus result. When the Laplacian potential is close to 0, the consistency of regional generation strategies is reached.

Q-learning is a value-based algorithm in reinforcement learning whose main idea is to build a Q-table with state–action value. Combined with the time difference (TD) method, the update formulation of Q-learning is described as:

$$\begin{aligned} &Q_{k+1}(s_k,a_k)=Q_k(s_k,a_k) \\ &+\alpha\begin{bmatrix} r(s_k,a_k,s_{k+1}) \\ +\gamma\max_{a'\in A_a}Q_k\left(s_{k+1},a'\right)-Q_k(s_k,a_k)\end{bmatrix} \end{aligned} \tag{8}$$

where $Q_k(s_k,a_k)$ is the state–action value at time slot k. s_k and a_k are the state and action at time slot k, respectively. $r(s_k,a_k,s_{t-1})$ is the immediate reward function of transferring the state s_k to s_{k+1} by applying action a_k. a and γ are the learning factor and discount factor, respectively. a' is the action which can obtain the maximum Q value at the state s_{k+1}.

QD-learning adopts the GLP-based consensus algorithm in the update process of Q-learning (Kar et al., 2013). The update formulation can be rewritten as:

$$Q_{k+1}^i(s_k,a_k)=Q_k^i(s_k,a_k)-C_Q^i(s_k,a_k)+I_Q^i(s_k,a_k) \tag{9}$$

where $C_Q^i(s_k,a_k)$ is the consensus potential to represent the collaboration with neighbors of the agent i. $I_Q^i(s_k,a_k)$ is the innovation potential to represent the learning process of the agent i. Base on Eqs. (8) and (9), the consensus potential and innovation potential can be described as:

$$C_Q^i(s_k,a_k)=\beta_k(s_k,a_k)\sum_{j\in\Omega}\left(Q_k^i(s_k,a_k)-Q_k^j(s_k,a_k)\right) \tag{10}$$

$$I_Q^i(s_k,a_k)=\alpha_k(s_k,a_k)\begin{bmatrix} r(s_k,a_k,s_{k+1}) \\ +\gamma\max_{a'\in A_u}Q_k^i\left(s_{k+1},a'\right) \\ -Q_k^i(s_k,a_k)\end{bmatrix} \tag{11}$$

where $\beta_k(s_k,a_k)$ and $\alpha_k(s_k,a_k)$ are the consensus weight and innovation weight at kth iteration, which can be calculated as:

$$\beta_k(s_k,a_k)=\frac{p_2}{\left(N_p(s_k,a_k)+1\right)^{\tau_2}} \tag{12}$$

$$\alpha_k(s_k,a_k)=\frac{p_1}{\left(N_p(s_k,a_k)+1\right)^{\tau_1}} \tag{13}$$

where $N_p(s_k,a_k)$ is the sampling number, which is equal to the batch size of sampling from replay buffer in Q-learning algorithm. p_1, p_2, τ_1 and τ_2 are the hyper-parameters of QD-learning, which determine the algorithm performance.

3. Proposed framework of BELIEFS

3.1. Structure of two-levels blockchain

Considering the hierarchical, federated system structure in a MLPS, a two-levels blockchain structure shown in Fig. 1, is applicable for multiple agents to conduct the ATC cooperatively.

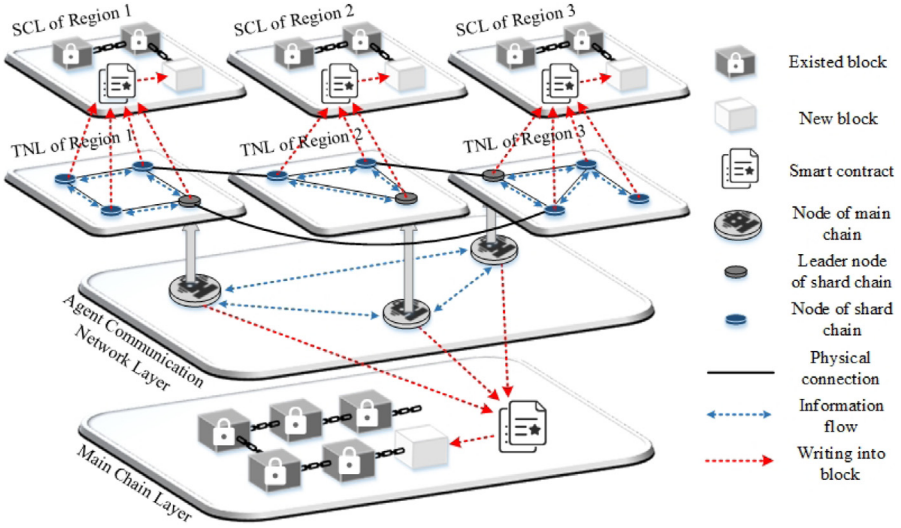


Fig. 1. The structure of the two-levels blockchain for a three-regions MLPS.

In each sub-region of a MLPS, a PV-bus node, which connected to generators, is selected to deploy an agent with QD-Learning Algorithm. We build a main chain with PV-buses as the nodes, and respectively build several shard chains with the PQ-buses as the nodes. Notice that the slack bus will be regard as a node of shard chain in its sub-region. Therefore, the PV-buses participate in both the cooperation between agents on the main chain and the data interaction between nodes on their shard chains.

The purpose of two-Levels blockchain proposed in this paper is to ensure the authenticity of data interaction and the effectiveness of collaboration in the process of agent training. On the main chain, considering the consensus term in Eq. (10) in the Q value calculation of each agent, the data interaction between multiple agents is the Q values of all agents. Each agent will update its own Q value based on Eq. (9) according to the acquired Q values of other agents. On each shard chain, the data interaction between topological nodes of the sub-regions is the states of all topological nodes data. After the consensus of blockchain, the data will be aggregated and stored by the leader node of each sub-region, so as to facilitate the agent to obtain the necessary observation.

3.2. Data flow of the proposed intelligence paradigm

The proposed intelligence paradigm mainly relies on the direct interaction with environment to tackle complex tasks of a MLPS as shown in Fig. 2. Through the adjacent matrix of the communication network topology of the region, each agent will obtain its partial observations from the state matrix, including the measure data of the region, the historical actions of the agent and the cooperative actions of other agents.

However, the system states of MLPS vary greatly under different operating scenarios, which makes it difficult for the agent to quickly find the optimal strategy in the highly redundant action space. Therefore, a dueling DQN is adopted, which divides Q function into state function and advantage function to accurately reflect state value and advantage value respectively. Through several hidden layers of full connection, the final output is calculated as follows:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \begin{pmatrix} A(s,a;\theta,\alpha) - \\ \sum_{a'} A(s,a';\theta,\alpha)/|A| \end{pmatrix} \tag{14}$$

where, $V(s;\theta,\beta)$ is state function, whose output is a scalar. $A(s,a;\theta,\alpha)$ is advantage function, which output a vector whose length is equal to the size of the action space. θ denotes the parameters of the M hidden layers. α and β denote the parameters of value function layer and advantage function layer, respectively.

After the data flowed out from the neural network, the action, which is mapping of the neural network output in the action space, is conducted in the environment. Finally, the states matrix will be updated for the data flow in the next interaction.

3.3. Training algorithm

Based on the QD-learning algorithm introduced in Section 2.3, we propose a training algorithm to improve the learning efficiency of the agents. The pseudo-code of the training algorithm is shown in Algorithm 1. In the beginning, the training dataset will be shuffled to ensure both the randomness and completeness of the case selection.

In each step of an episode, before each agent selects their actions, a procedure is conducted to check the illegality of all actions of each agent in its action space. The illegality judgment of an action is based on the constraints in the model (1), which will return true if the action cannot satisfy the constraints of MLPS. For each agent, the illegal actions will be not selected in this step. This procedure will help agents reduce their action space to efficiently select effective actions in each step. After that, we convert the actions of multi-agents to a joint action, and then execute the joint action in the grid environment. The next states, reward and done flag will be returned by the grid environment. The trajectory of state–action–reward–state will be stored by multi-agents in the replay buffer as an experience. After the size of replay buffer reach certain number, a batch of experiences are sampled to training the neural networks of the agents.

A double DQN is applied to prevent from overestimating in the training process. Based on Eqs. (10) and (11), the consensus term and innovation term can be calculated under the double DQN structure as follow:

$$C_Q^i(s_k,a_k) = \beta_k(s_k,a_k) \sum_{j=\Omega} \begin{pmatrix} Q_k^{i,main}(s_k,a_k) \\ -Q_k^{j,\text{main}}(s_k,a_k) \end{pmatrix} \tag{15}$$

$$I_Q^i(s_k,a_k) = \alpha_k(s_k,a_k) \begin{bmatrix} r(s_k,a_k,s_{k+1}) \\ +\gamma \max_{a\in A_a} Q_k^{i,tar}(s_{k+1},a') \\ -Q_k^{i,main}(s_k,a_k) \end{bmatrix} \tag{16}$$

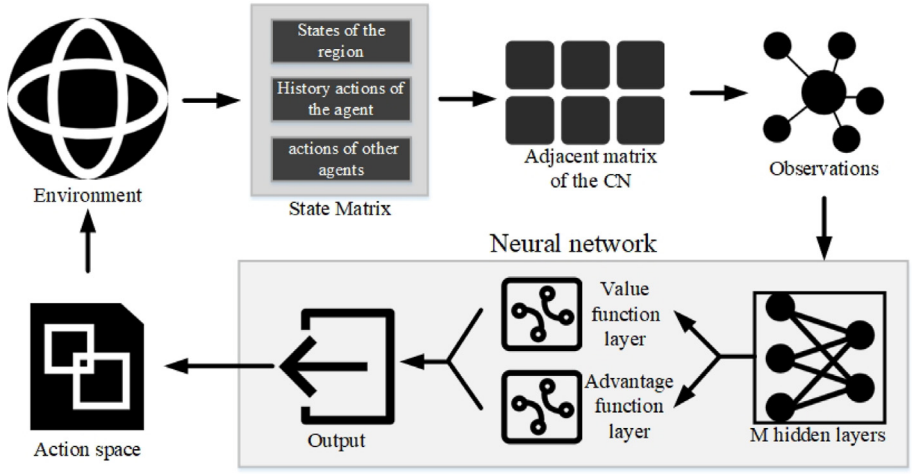


Fig. 2. The schematic diagram of the data flow.

where $Q_k^{i,main}$, $Q_k^{i,tar}$ are the output of the main DQN and the target DQN, respectively. Therefore, the loss function can be described as:

$$L(\theta) = E_{(x_2,a_k)-p}\left[\left(C_Q^i\left(s_k,a_k;\theta,\alpha,\beta\right)\right)^2\right] + E_{(s_z,A_k)\cdots p}\left[\left(I_Q^i\left(s_k,a_k;\theta,\alpha,\beta\right)\right)^2\right] \tag{17}$$

By minimizing a sequence of loss function drawn from the replay buffer, the parameters of the dueling DQN can be updated. Note that, to avoid the algorithm falling into local optimum, a soft update method is adopted, which is

$$\hat{\theta} \leftarrow \zeta\theta + (1-\zeta)\hat{\theta} \tag{18}$$

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.

Algorithm 1 Training structure for solving the control problem of a MLPS.

```
Initialize: neural network for each agent with random parameters, the environment of MLPS, training dataset, max case numbers M, discount factor, initialize replay buffer;
for episode = i to I do
    shuffle the training data and set the case id = 0;
    reset the environment, obtain the initial states s_t;
    for t = 1 to T do
        obtain the partial observations O_t^i, i ∈ n.
        for each agent do
            check the illegality of all actions in the grid;
            select the optimal action besides illegal actions;
        end for
        convert the actions to a joint action;
        conduct the joint action in the grid environment;
        obtain the next states, reward and done;
        if done then
            break
        end if
        add experience of the agents into the replay buffer;
    end for
    sample a batch of experience to update dueling DQN
    case id = case id + 1 if case id < M else 0
end for
```

3.4. Attack-tolerant capability

The offline training process of DDRL is unsupervised and time-consuming. The malicious attacks, which will be disturbed or even destroy the effectiveness of agents training process, are hardly been timely detected during the training process. Considering the four kinds of attack-induced faults mentioned in system model (1), the main malicious attacks for a MLPS are considered as the following two typical types (Yan and Xu, 2020b):

(1) Interaction attacks: these attacks will disturb the data interaction in each region, which may cause faults of data collection and lead to the abnormal of the agent execution;
(2) Cooperation attacks: these attacks will disturb the cooperation among multiple agents, which may destroy network communication and disturb the cooperative behaviors of multi-agents.

In the case of an Interaction Attack, if it cannot conduct a 51% attack on the shard chains, the agent can still obtain the actual network state data through the Byzantine fault tolerance mechanism of the blockchain to make the correct decision. When parts of shard chains tamper with over 51% attack, according to the model (1), the faults caused by the interaction attacks will make the agents of these regions unable to obtain authentic and effective system state. According to Eq. (9), the Q value update of each agent will depend on the consensus term C_Q. That indicates the agents under attack will transmit the wrong Q value to disturb the calculation of the consensus term of other agents.

In the case of cooperation attacks, according to the model (1), it can block communication and cooperation between agents. Under the attacks, however, each agent on the main chain will not be able to obtain the true Q value of other agents. Attackers will make multiple agents difficult to have an ineffective joint strategy. After conducting this joint strategy in the MLPS, the agents cannot get stable positive rewards. Even if the attackers collude to keep the consensus term of all agents within the normal range, the innovation term I_Q of each agent in training process will become abnormal, which result in the training loss value of multiple agents fluctuate with the abnormality.

The malicious attacks will generate batches of bad experience to prevent the agents from effectively learning, which will last for a period of training time even after the malicious attack is cleared. The attack-tolerant capability discussed in this paper is presented as follows: (1) In the case of less than 51% attacks, blockchain can prevent data tampering and other data interaction faults caused by malicious attacks; (2) After any shard chains are attacked by

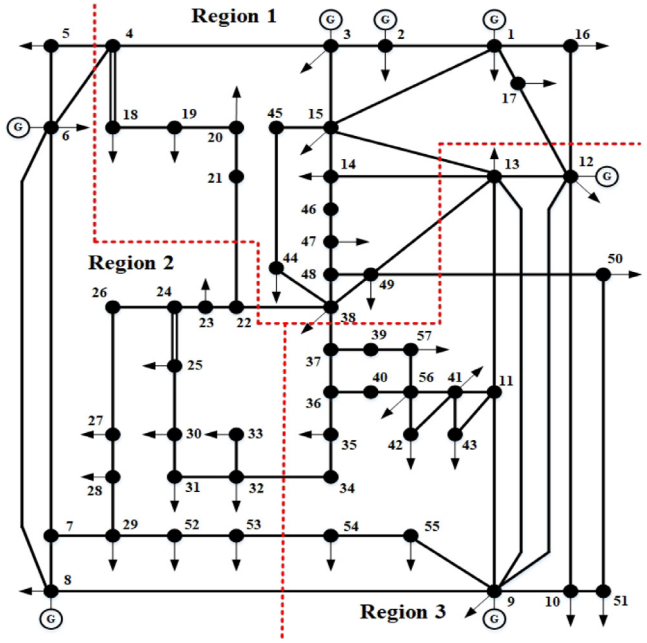


Fig. 3. The topology of the modified IEEE 57-bus system with 3 regions.

51% attack, the malicious attacks can be detected by monitoring the consensus terms of all agents. After the malicious attack is cleared, multiple agents can be trained effectively through algorithm 1 to overwrite the bad experience. (3) After the main chain is attacked by 51% attack, the malicious attacks can be detected by monitoring the innovation terms of all agents even in the case of collusion by attackers. Similarly, after the malicious attack is cleared, effective training can be carried out again for multiple agents using algorithm 1 to overwrite the bad experience. This attack-tolerant capability will be fully verified in the following case studies.

4. Case study

4.1. Simulation setup

In this section, a modified IEEE 57-bus system with 3 regions is used to evaluate our proposed method, which is shown in Fig. 3. The slack bus of this system is bus-1, which means Gen-1 is not under the control of Region 2. All the load scenarios in the training dataset are perturbed with a random range from 80% to 120% of a baseline load scenario (see Fig. 3).

Based on the objective and constraints of The tertiary control problem, the reward function is designed as

$$r = \begin{cases} R_{\text{div}} & \text{if diverged} \\ R_{rho} \sum_{l \in \Omega_L} \left| P_l - 0.93 \times P_l^{\max} \right| & \text{if abnormal} \\ R_r \sum_{k \in Q_i} C_k + R_c & \text{if normal} \end{cases} \tag{19}$$

where R_{div} is a negative constant corresponding to the divergence of power flow. R_{rho} are the negative coefficients corresponding to heavy load case. R_r and R_c are the negative coefficient and positive constant corresponding to the total generation cost, respectively. P_l and $P_l^{\max}$ are transmission active power and transmission capacity of line l, respectively. C_k is the generation cost of generator k. Ω_B, Ω_L and Ω_G are set of buses, lines and generators, respectively.

For better training our agents, we develop an offline power grid environment to interact with the agents, which consists of a power flow solver based on PyPower and several utility functions. The work flowchart of ACOPF grid environment interacting with agents in an episode is shown in Fig. 4. In each episode, we use a "reset" function to initialize a load scenario of a month and obtain the states of the power grid. The load scenario is randomly selected from a load dataset, which contains every 15 min load data of one year. After agents perform their actions based on their partial observations, the grid environment will conduct the calculation of AC power flow in PyPower. The next states of the power system and reward will be returned by the grid environment. Moreover, we will set a "done" flag to determine whether a load scenario is finished or not. Generally, there are two situations making the load scenario finish: (1) the result of power flow calculated by PyPower solver is divergent; (2) agents have run through all the load cases of a load scenario. At each step of an episode, the agents will replace the states with the next states to prepare for the next step. The episode will terminate while the "done" flag is true or the episode steps reach the maximum episode length.

Table 1
The ramping capacity of generators.

Gen ID	Ramping (MW/15 min)	Generation capacity (MW)
1	100	575.88
2	50	100
3	65	140
6	50	100
8	90	550
9	60	100
12	60	410

Table 2
The hyper-parameters used in QD-learning.

Parameters	Value	Parameters	Value
γ	0.99	R_{rho}	−1
p_1	0.2	R_w	−0.01
p_2	0.8	R_c	100
τ_1	0.99	R_{div}	−1000
τ_2	0.49		

The ramping capacity information of generators is presented in Table 1, and all the hyper-parameters (Zhang et al., 2018a) of QD-learning algorithm is shown in Table 2. The two dueling DQN is composed of two hidden layers with 256 neurons. In the hidden layers, we choose leaky rectified linear unit (Leaky RELU) as the activation function. The batch size is set to 64 and the batch normalization (BN) can be applied to the inputs. The agents take last 5 recent history observations of the power grid states. Therefore, the last 5 recent observation states are stacked with the current observations and used as input for DDQN. The maximum number of steps in an episode is set to 96, which is the maximum total time slots of load scenarios in a day. The simulation is conducted by PyPower 5.1.4/Pytorch-GPU Cuda 11.0 using a Linux server with NVIDIA GPUs.

4.2. Learning capability

In this part, we adopted Algorithm 1 proposed in Section 3.3 to train our multi-agents for 500 000 steps. In order to clarify the impact of consensus method on multi-agent training process in QD-learning, two schemes are set in this paper as: (1) Scheme 1 trains the agents with consensus; (2) Scheme 2 trains the agents with no consensus. The experiments of the two schemes are conducted under the same hyper-parameters and random seeds. The loss curves and reward curves of the agents during the training process, shown in Fig. 5, are checked to evaluate the convergence performance of our proposed method.

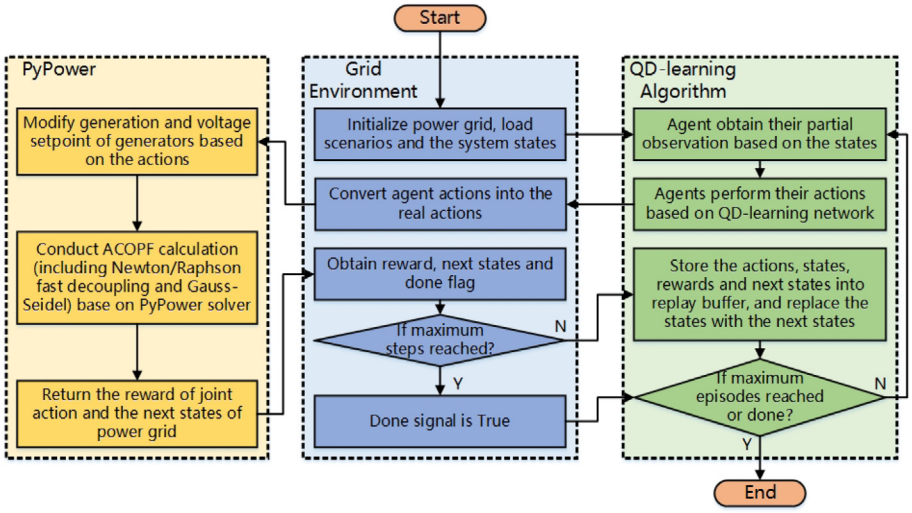
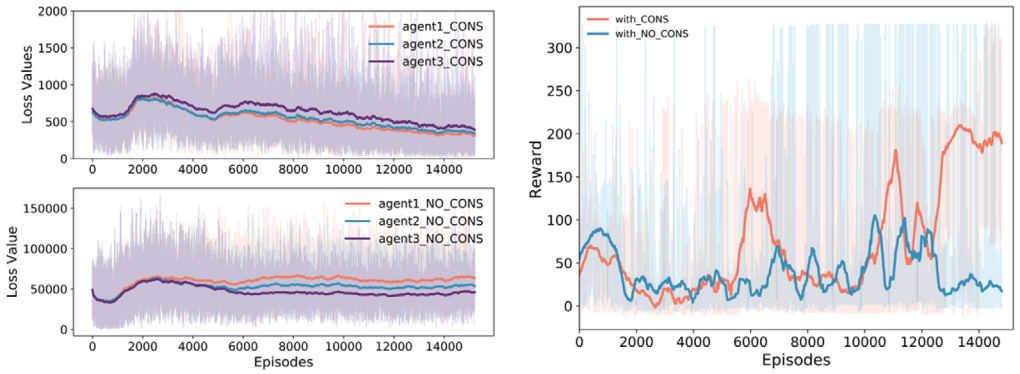


Fig. 4. Flowchart of grid environment interacting with agents in an episode.



(a) The loss curves. (b) the reward curves.

Fig. 5. The moving average curves of different schemes in training.

As shown in Fig. 5(a), the legend of "agent_Scheme 1" denote the consensus method is adopted in the training process, and the legend of "agent_Scheme 2" denote the consensus method is not adopted. The loss values of the agents with consensus decrease progressively and finally converge to around 300. However, the loss values of the agents in Scheme 2 are still in a relatively high range. Through interacting with other agents in the consensus process, each agent in Scheme 1 can obtain global information through and turn its decision-makings behavior into a stationary Markov process. The moving average reward curve with the legend of "Scheme 1" demonstrates that the joint strategy of MAS in Scheme 1 converge to an effective solution of ACOPF.

Notice that the fluctuations of Scheme 2 are at a magnitude of 10^4, this is because that the training with no consensus is conducted in a dynamic Markov environment. Although the overall trend of the loss curves of Scheme 2 seems come to convergence, without consensus method, the agents hardly reach an effective Nash equilibrium due to lacking of the global information about the grid environment, which is shown in Fig. 5(a).

4.3. Executing performance

In this part, we adopt a test scenario whether the trained multiple agents can solve the problem of the tertiary control. Therefore, a scenario of 96 load cases, which contains every 15 min load data of a day, is randomly selected to evaluate the execution performance of multiple agents. The agents need to make decisions of the generation outputs for the time-sequential 96 load cases, where the constraints of generation ramping are considered in execution process. As the learning capability of the Scheme 1 is relatively poor and inefficient, we focus on the execution performance of the experiments in Scheme 2 at this section. In addition, the Interior Point Method (IPM) are adopted to solve the ACOPF problem for making a comparison.

Table 3
The average calculation times of different algorithm.

Algorithm	Time (s)
Proposed Algorithm	0.14
IPM	1.94

It can be seen from Fig. 6 that the agents of our proposed Algorithm 1 can consecutively solve the M-ACOPF problem for the time-sequential 96 load scenarios under the constraints of generation ramping and line transmission capacity. By comparing the solutions of the different algorithms in Fig. 6, it can be found that the trend of active power output curves are generally the same. In addition, we can notice that the solution curve of IPM

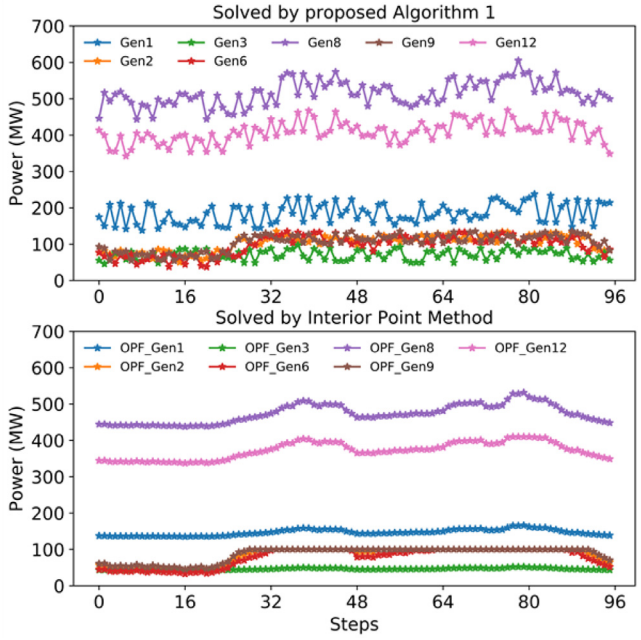


Fig. 6. The ACOPF solutions of different algorithms in test scenario.

is much smoother than that of Algorithm 1. It is because of the high dimension of the large discrete action space make it difficult for the agents to learn the most efficient strategies. However, as shown in Table 3, the calculation time of the agents trained by the proposed Algorithm 1 is much shorter than that of the IPM, which is more practical in the tertiary control.

Furthermore, we monitored the probability of selecting each action of the agents to figure out what strategies they have learnt. Considering that the action space of each agent is a symmetric discrete space with the range of [−1,1], the action with the Action ID in the middle indicates that the agents will slightly adjust the active power output of the generators. In Fig. 7, the whole training process is divided into four stages. After sampling 3000 action choices of the agent in each stage, there are four layers in each subgraph, with light to dark colors, representing the probability distribution of the agent's action selection from the initial stage to the final stage. As shown in Fig. 7(a), the agents in Scheme 1 have learnt more clear and effective strategies than the agents in Scheme 2. The strategy learned in Scheme 1 is more likely to select actions in the middle of the action space, while the strategy learned in Scheme 2 is more likely to adjust the generator of Agent 3 frequently and dramatically. In addition, it can be seen from Fig. 7(b) that the four layers of Agent 1 are basically coincident, while the dark layers of Agent 2 and Agent 3 are distributed divergently. It is because of the environment of Scheme 2 is dynamic, such that the trained agents either fall into local convergence without action exploration, or fail to converge to effective strategies during the process of action exploration.

4.4. Validation of attack-tolerant capability

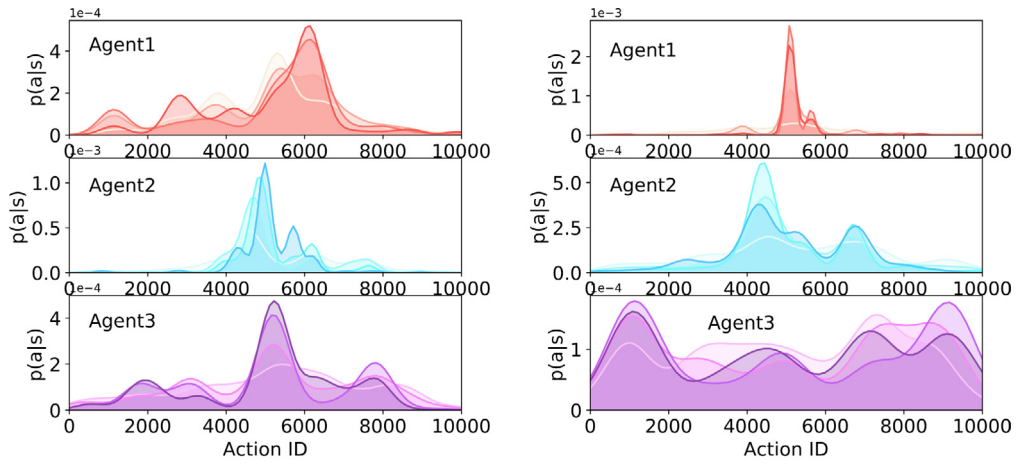
As described in Section 3.4, the abnormality of the training process under malicious attacks can be detected by monitoring the consensus term C_Q and innovation term I_Q. In this section, we present the attack-tolerant capability through monitoring the loss values, the consensus terms and the innovation terms under different malicious attacks.

Firstly, we set an attack on the main chain, which starts at around 20 000 steps in the training process and can destroy the consensus between the agents. Therefore, agents cannot obtain the Q value of other agents, which indicate that the consensus terms of all agents come close to 0. As shown in Fig. 8(a), the duration of the attack is signed by the red dash circle. Assumed that after 5 min, the attack is detected and cleared by monitoring the abnormality of the consensus terms and innovation terms. As the malicious attacks last for 5 min, the agents have stored batches of bad experience, which make the loss values of the agents greatly increased. It can be seen that the impact of the malicious attack on loss values last for almost 4000 episodes. After that, through regular training based on Algorithm 1, the loss values of the agents begin to decrease and gather. Finally, the loss values can converge to the original values after about 12 000 episodes, which is nearly 3 h of training.

Similarly, an attack is set on the shard chain of region 2 in the training process. The attack also starts at around 20 000 steps, which falsifies the data observed by the agent 2 such that the calculated Q value is incorrect and uncertain. It can be seen in Fig. 8(b), as the agent 1 and agent 3 are in the regular states, the fluctuations of the loss values, consensus term and innovation term are smaller than the case of attacking the main chain. However, the loss values converge to the original values after around 8000 episodes, which is nearly 2 h of training. To demonstrate the effectiveness of the consensus mechanism, we set up a scenario that the agents will be trained by Scheme 2 after malicious attacks occurred. To make complete comparison, the duration of the attacks is also set as 5 min. It can be seen from Figs. 9(a) and 9(b), as each agent do not communicate with others, the loss values will not increase sharply without consensus such that the malicious attacks cannot be easily detected. However, the loss curves of the agents in Scheme 2 seem to keep fluctuating after the attacks occurred. The loss curves cannot reach a convergence, without consensus, which result in worst rewards of the agents shown in Fig. 10.

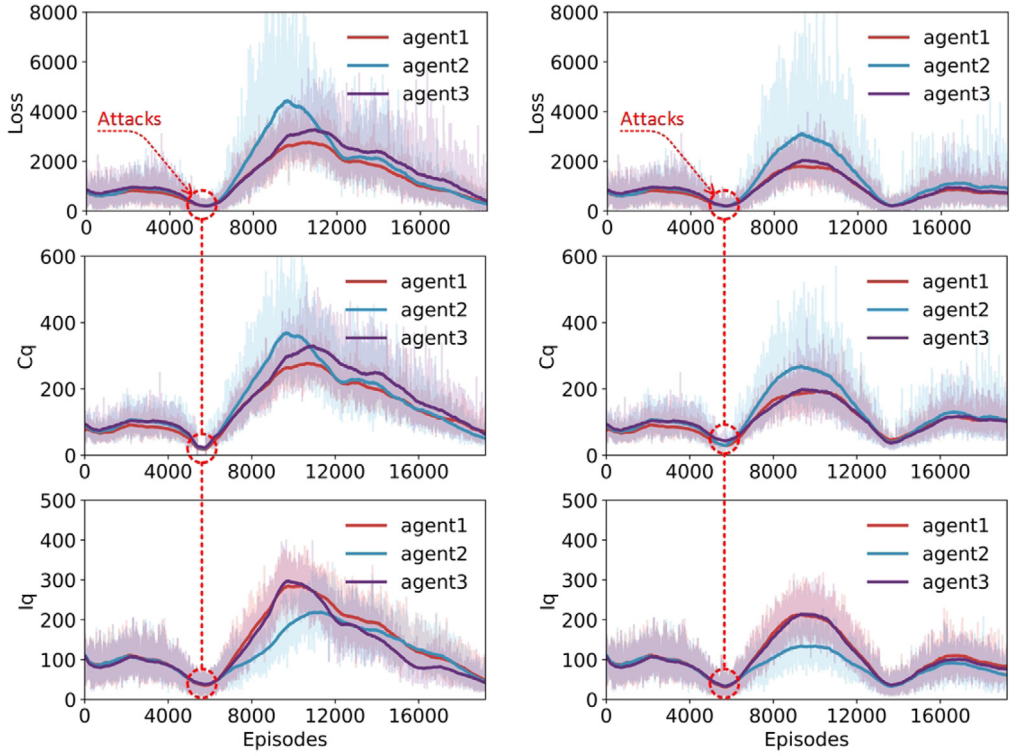
5. Conclusion

DRL is becoming a significant tools for solving the high-dimensional, nonlinear control problems and has widely applied in the field of MLPS operation control and dispatch. However, the training phase of deep reinforcement learning agent is a time-consuming process, which would be invalid in face of some data fault or agent fault occurring in the training phase. When deep reinforcement learning is used for the distributed control of a multi-regional power system, it is easily to suffer the malicious attacks in the training and execution, which reduces the practicability of deep reinforcement learning in the field of multi-regional power grid operation control and dispatch. Therefore, this paper proposes a blockchain-enabled attack-tolerant distributed deep reinforcement learning-based intelligence paradigm. Through the framework of blockchain and the consensus mechanism in QD-learning algorithm, the attack-tolerance of agents are effectively increased in the training and execution phase. This paper provides a basic paradigm for establishing a trusted power grid operation control and dispatch platform. In our future works, more deep reinforcement learning algorithms will be combined with the consensus mechanism of QD-learning to improve the control performance.



(a) Scheme 1. (b) Scheme 2.

Fig. 7. The action probability of the agents in different schemes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Attacks on main chain. (b) Attacks on shard chains.

Fig. 8. The moving average curves of the agents under different malicious attacks in Scheme 1.

CRediT authorship contribution statement

Siyuan Chen: Conceptualization, Methodology, Software, Writing – original draft, Supervision. **Jun Zhang:** Conceptualization, Writing – review & editing, Funding acquisition. **Yuyang Bai:** Software, Validation, Formal analysis, Data curation, Visualization. **Peidong Xu:** Validation, Formal analysis. **Tianlu Gao:** Writing – review & editing. **Huaiguang Jiang:** Writing – review & editing. **Wenzhong Gao:** Writing – review & editing. **Xiang Li:** Data curation.

Declaration of competing interest

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

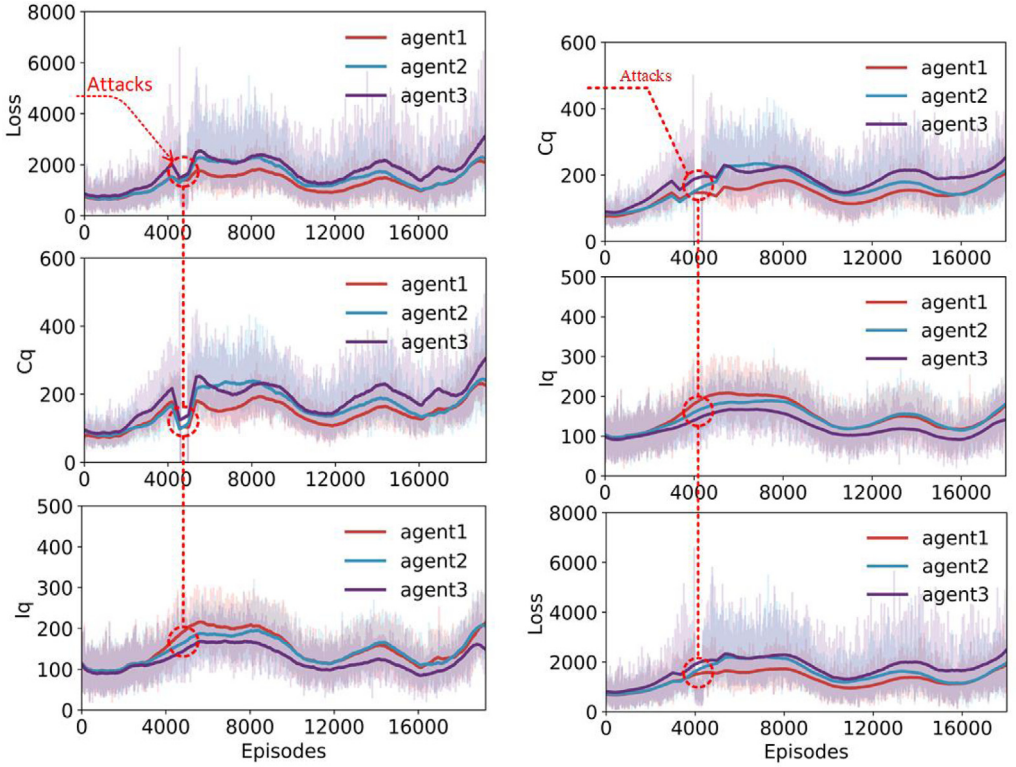


Fig. 9. The moving average curves of the agents under different malicious attacks in Scheme 2.

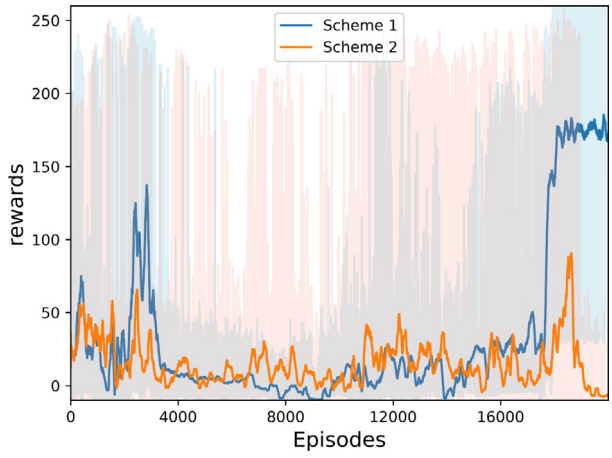


Fig. 10. The moving average reward curves of training under malicious attacks in different schemes.

Acknowledgment

This work was supported by the National Key R&D Program of China under Grant 2018AAA0101504.

References

Ahmed, M.M., Hasan, M.K., Shafiq, M., Qays, M.O., Gadekallu, T.R., Nebhen, J., Islam, S., 2021. A peer-to-peer blockchain based interconnected power system. Energy Rep. http://dx.doi.org/10.1016/j.egyr.2021.08.071.

Chen, Y., Lu, J., Yu, X., Hill, D.J., 2013. Multi-agent systems with dynamical topologies: Consensus and applications. IEEE Circuits Syst. Mag. 13 (3), 21–34. http://dx.doi.org/10.1109/MCAS.2013.2271443.

Chen, C., Zhang, K., Ni, M., Wang, Y., 2021. Cyber-attack-tolerant frequency control of power systems. J. Mod. Power Syst. Clean Energy 9 (2), 307–315. http://dx.doi.org/10.35833/MPCE.2019.000185.

Diao, R., Wang, Z., Shi, D., Chang, Q., Duan, J., Zhang, X., 2019. Autonomous voltage control for grid operation using deep reinforcement learning. In: 2019 IEEE Power Energy Society General Meeting (PESGM). pp. 1–5. http://dx.doi.org/10.1109/PESGM40551.2019.8973924.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P., 2016. Benchmarking deep reinforcement learning for continuous control. In: International Conference on Machine Learning. PMLR, pp. 1329–1338.

Duan, J., Shi, D., Diao, R., Li, H., Wang, Z., Zhang, B., Bian, D., Yi, Z., 2020. Deep-reinforcement-learning-based autonomous voltage control for power grid operations. IEEE Trans. Power Syst. 35 (1), 814–817. http://dx.doi.org/10.1109/TPWRS.2019.2941134.

Foti, M., Mavromatis, C., Vavalis, M., 2021. Decentralized blockchain-based consensus for optimal power flow solutions. Appl. Energy 283, 116100.

Gu, S., Holly, E., Lillicrap, T., Levine, S., 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 3389–3396. http://dx.doi.org/10.1109/ICRA.2017.7989385.

Haoran, D., Ming, Y., Fang, C., Guozhong, S., 2015. Reactive power and voltage optimization control approach of the regional power grid based on reinforcement learning theory. Trans. China Electrotech. Soc. 30 (12), 408–414.

Huang, Q., Huang, R., Hao, W., Tan, J., Fan, R., Huang, Z., 2020. Adaptive power system emergency control using deep reinforcement learning. IEEE Trans. Smart Grid 11 (2), 1171–1182. http://dx.doi.org/10.1109/TSG.2019.2933191.

Hug, G., Kar, S., Wu, C., 2015. Consensus + innovations approach for distributed multiagent coordination in a microgrid. IEEE Trans. Smart Grid 6 (4), 1893–1903. http://dx.doi.org/10.1109/TSG.2015.2409053.

Jin, X., Lü, S., Deng, C., Chadli, M., 2021. Distributed adaptive security consensus control for a class of multi-agent systems under network decay and intermittent attacks. Inform. Sci. 547, 88–102.

Kar, S., Moura, J.M.F., Poor, H.V., 2013. $\mathcal{QD}$-Learning: A collaborative distributed strategy for multi-agent reinforcement learning through Consensus + Innovations. IEEE Trans. Signal Process. 61 (7), 1848–1862. http://dx.doi.org/10.1109/TSP.2013.2241057.

Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P., 2021. Deep reinforcement learning for autonomous driving: A survey. IEEE Trans. Intell. Transp. Syst. 1–18. http://dx.doi.org/10.1109/TITS.2021.3054625.

Li, Y., Wang, R., Yang, Z., 2021. Optimal scheduling of isolated microgrids using automated reinforcement learning-based multi-period forecasting. IEEE Trans. Sustain. Energy 1. http://dx.doi.org/10.1109/TSTE.2021.3105529.

Li, J., Yu, T., 2021. A new adaptive controller based on distributed deep reinforcement learning for PEMFC air supply system. Energy Rep. 7, 1267–1279. http://dx.doi.org/10.1016/j.egyr.2021.02.043.

Lin, K., Zhao, R., Xu, Z., Zhou, J., 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1774–1783.

Liu, H., Yu, C., Wu, H., Duan, Z., Yan, G., 2020. A new hybrid ensemble deep reinforcement learning model for wind speed short term forecasting. Energy 202, 117794. http://dx.doi.org/10.1016/j.energy.2020.117794.

Qin, J., Ma, Q., Shi, Y., Wang, L., 2017. Recent advances in consensus of multi-agent systems: A brief survey. IEEE Trans. Ind. Electron. 64 (6), 4972–4983. http://dx.doi.org/10.1109/TIE.2016.2636810.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S., 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 4295–4304.

Sanaye, S., Sarrafi, A., 2021. A novel energy management method based on Deep Q Network algorithm for low operating cost of an integrated hybrid system. Energy Rep. 7, 2647–2663. http://dx.doi.org/10.1016/j.egyr.2021.04.055.

Su, Q., Khan, H.U., Khan, I., Choi, B.J., Wu, F., Aly, A.A., 2021. An optimized algorithm for optimal power flow based on deep learning. Energy Rep. 7, 2113–2124. http://dx.doi.org/10.1016/j.egyr.2021.04.022.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al., 2017. Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296.

Tan, M., 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning. pp. 330–337.

Yan, Z., Xu, Y., 2020a. A multi-agent deep reinforcement learning method for cooperative load frequency control of a multi-area power system. IEEE Trans. Power Syst. 35 (6), 4599–4608. http://dx.doi.org/10.1109/TPWRS.2020.2999890.

Yan, Z., Xu, Y., 2020b. Real-time optimal power flow: A Lagrangian based deep reinforcement learning approach. IEEE Trans. Power Syst. 35 (4), 3270–3273. http://dx.doi.org/10.1109/TPWRS.2020.2987292.

Yang, Q., Wang, G., Sadeghi, A., Giannakis, G.B., Sun, J., 2020. Two-timescale voltage control in distribution grids using deep reinforcement learning. IEEE Trans. Smart Grid 11 (3), 2313–2323. http://dx.doi.org/10.1109/TSG.2019.2951769.

Zhang, X.S., Li, Q., Yu, T., Yang, B., 2018a. Consensus transfer Q-learning for decentralized generation command dispatch based on virtual generation tribe. IEEE Trans. Smart Grid 9 (3), 2152–2165. http://dx.doi.org/10.1109/TSG.2016.2607801.

Zhang, J., Lu, C., Si, J., Song, J., Su, Y., 2018b. Deep reinforcement leaming for short-term voltage control by dynamic load shedding in china southem power grid. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Zhou, Y., Zhang, B., Xu, C., Lan, T., Diao, R., Shi, D., Wang, Z., Lee, W.-J., 2020. A data-driven method for fast AC optimal power flow solutions via deep reinforcement learning. J. Mod. Power Syst. Clean Energy 8 (6), 1128–1139. http://dx.doi.org/10.35833/MPCE.2020.000522.