Received April 14, 2021, accepted May 15, 2021, date of publication May 20, 2021, date of current version June 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3082259

Deep Reinforcement Learning-Based Service-Oriented Resource Allocation in Smart Grids

LINHAN XI[1], YING WANG[1], (Member, IEEE), YANG WANG[2], ZHIHUI WANG[2], XUE WANG[1], AND YUANBIN CHEN[1]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]China Electric Power Research Institute Company Ltd., Beijing, China

Corresponding author: Ying Wang (wangying@bupt.edu.cn)

This work was supported by the Smart Grid Technical Project "Evolution of Power Wireless Network and Application Analysis of 4G and 5G Technologies" under Grant 5700-201955234A-0-0-00.

ABSTRACT Resource allocation has a direct and profound impact on the performance of resource-limited smart grids with diversified services that need to be timely processed. In this paper, we investigate a joint communication, computing, and caching resource allocation problem with distinct delay requirement of services in smart grids. This paper aims to optimize the long-term system utility based on reward and loss function. Considering the unknown dynamic environment as well as the huge state and action space in smart grids, a deep reinforcement learning algorithm based on the polling method is exploited to learn the policy by interacting with the environment. Specifically, the edge nodes (ENs) act as agents to enable the services to schedule resources appropriately. Then, the agents that are allocated based on the service requirements are queried according to the polling mechanism and the well-designed reward function is utilized to update the strategy. Extensive simulation results show that the proposed algorithm outperforms three known baseline schemes in terms of network performance with decision results. Besides, in the face of a large number of services in the smart grids, the proposed system still surpasses that of existing several baseline schemes, especially in the improvement of cache hit rate and the decrease of computing delay.

INDEX TERMS Smart grids, edge computing, deep reinforcement learning, resource allocation.

I. INTRODUCTION

The Internet of Things (IoT) is an emerging domain dedicated to connecting ubiquitous objects to the Internet, and the number of connected devices will reach 28 billion by 2021 [1]. With the improvement of IoT requirements, the grid pattern is also changing, and the distributed concept is forcing the traditional grids to adapt to the new situation [2]. The smart grids have replaced traditional networks by using distributed power control and communication technologies (such as 5G) to improve operational efficiency [3]. The distributed smart grids integrate many IoT devices and upload information to the Internet in time to avoid problems such as failures and capacity limitations. Many service concepts have been introduced into the smart grids, including smart meters (SMs), advanced metering infrastructure (AMI), distributed generators (DG), and so on [4]. The service communication network can be represented by a hierarchical multi-layer architecture, and the architecture can be composed of user local area network (IAN), neighborhood network (NAN), and wide area network (WAN) based on data rate and coverage [5].

In smart grids, synchronous power grid monitoring tends to operate with high precision to realize real-time fault monitoring. However, the stringent delay requirements raise significant challenge to the communication infrastructure [6]. Some mission-critical applications have tight delay constraints, such as the distribution automation deployed in substation requires information transmission within 4ms. However, some smart meters send data at long intervals, such as 15 minutes [7]. In order to reduce service delay in response to service-oriented resource allocation in smart grids, edge computing (EC) and deep reinforcement learning (DRL) are introduced.

A. SMART GRIDS AND EDGE COMPUTING

EC provides a distributed paradigm for computing and caching in smart grids by deploying servers at the edge. In order to reduce the burden from more and more power

The associate editor coordinating the review of this manuscript and approving it for publication was Amedeo Andreotti.

devices, edge nodes (ENs) have been given the ability to perform computing and caching at the edge by allowing services to be handled at edge servers distributed close to users [8]. A multi-layer radio access network in the cloud is designed and a cooperative resource allocation algorithm is proposed to reduce service delays in edge networks and optimize throughput [9]. The heuristic algorithm [10], greedy algorithm [11] and game theory [12] are applied to optimize the allocation resource in smart grids.

However, there are still many problems to be tackled about EC, such as 1) which EN handles tasks; 2) whether EN offloads tasks to the cloud or receives tasks from the cloud; 3) how to allocate resources for these tasks in smart grids. In the resource allocation problem of EN, it is not only necessary to optimize communication, computing, and caching resources, but also to select the appropriate EN, so many researchers describe this problem as an NP problem. Q. He *et al.* believed that the edge user allocation (EUA) is NP-hard, so they proposed a game-theoretic approach to solve this problem in [13]. In order to meet the delay requirements of users, T. Ouyang *et al.* used Lyapunov optimization to decompose the complex problem into real-time optimization problems which were NP-hard and proposed a Markov approximation algorithm to solve the problem [14]. The state and action space increases exponentially with the number of the user's requests and the devices. In addition, due to the dynamics of the network environment, e.g., the contents of the request and locations of the devices, the resource allocation in smart grids can no more be tackled via the traditional one-shot optimization algorithms. Therefore, in the next subsection, DRL is applied to tackle this problem.

B. SMART GRIDS AND DEEP REINFORCEMENT LEARNING

With the expansion of smart grids scale and the rapid growth of the number of users, it has led smart grids to operate in more uncertain, complex environments. Due to the influence of uncertain factors, traditional methods cannot adapt to the development of smart grids and the requirements of the customers. On the other hand, the extensive deployment of AMI [15], WAMS [16] and power system nodes produces massive data, which can not only provide data base for DRL training, but also reduce the impact of uncertain factors. Due to the robust learning ability and interaction with the environment, DRL can collect information from a large amount of data and make adaptive decisions [17]. The information from users and devices is often not available in advance, but DRL can complete the decision to make to uninstall without prior experience [18]. Agent strategy can be divided into single agent strategy and multi-agent strategy. Single agent algorithm relies on an experienced replay buffer, such as Q-learning and DRL. In the multi-agent system, the agents update their policies in parallel and enable the use of replay buffers to train independent learners [19], and a single globally shared network is trained to output different policies for homogeneous agents [20]. Both cooperative multi-agent DRL [21] and non-cooperative methods [22] can be used to optimize resource allocation strategies.

The uncertainty and complex operation environment of smart grids bring challenges to the application of DRL. For example, different control methods and constraints of power devices make the model more complex, and there are multiple entities with different objectives in smart grids, which endows the reward functions with more difficulties [23]. In view of the above issues, multiple designs and modifications are required to adapt to different scenarios, which are as follows: (1) the reward function determines the efficiency of the algorithm, so it needs to be designed according to the actual problem; (2) information sharing between the state space and the behavior space determines the efficiency of the decision-making; (3) the scheduling and updating strategies of the agent need to adapt to the service characteristics of the smart grids.

This paper focuses on the joint optimization of computing and caching resources allocation problems which makes the ENs have communication, computing, and caching capability to process services independently. A resource allocation model of smart grids based on EC is proposed. In this model, the service delay includes network transmission delay and computing delay. In order to reduce the service delay, an algorithm based on DRL is proposed to explore the optimal resource allocation strategy. The main technical contributions are summarized as follows:

- We first design an EC system framework with three layers in smart grids that includes the service layer, edge layer, and cloud layer, where the objective of the proposed framework is to minimize the total service delay and obtain the optimal resource allocation strategy under the varying service requirements.
- An agent polling update deep reinforcement learning (APUDRL) algorithm is proposed to optimize the communication, computing, and caching strategy. It combines the neural network with the DRL-based polling method and explores strategy with the designed environment. In the face of a large number of services with different delay requirements, such as millisecond, second, and minute, the constraints in the optimization function are mapped to the penalty factors in the reward function. The SumTree sampling method is used to improve the efficiency of data sampling, and the separation of value function and advantage function is used to improve the learning efficiency.
- The numerous services in smart grids bring huge cache pressure, so a cache update strategy considering popularity and cache time, namely popularity and cache time (PACT) is proposed to improve the cache hit rate. The content popularity represents the frequency of users' requests for the content, and the cache time represents the length of time content has been cached.
- The extensive simulations are performed under varying scenarios in the proposed EC system in order to verify

the effectiveness of the proposed system model and the algorithm APUDRL. The numerical simulation results show that the performance of algorithms outperforms that of three baseline algorithms by at least 72.85%, 61.65% and 58.84%. The cache hit rate of the PACT is also surpasses that of the two baseline schemes.

The remainder of the paper is as follows. The system model is introduced in Section II. Then, in Section III, the APUDRL algorithm is proposed. In Section IV, simulation results prove that the proposed algorithm has excellent efficiency and adaptability. Finally, the conclusion is drawn in Section V.

II. SYSTEM MODEL AND PROBLEM FORMULATION

The system structure is shown in Fig. 1, and an EC system framework with three layers in smart grids is considered. The service layer is composed of users and power devices, which is used to send and receive services. Based on a variety of smart grids use cases and selected standards, three services with delay tolerance of milliseconds, seconds and minutes are considered, which are expressed as $\mathcal{S}^{\text{ser}} = \{s_1^{\text{ser}}, s_2^{\text{ser}}, s_3^{\text{ser}}\}$. The required response time for wide-area monitoring and closed-loop transient stability control should be in the range of milliseconds (e.g., <0.1 s) and they are denoted as s_1^{ser}. The required response time for smart appliance and load control device should be in the range of seconds and they are s_2^{ser}, and automation application is s_3^{ser} [5].

In this paper, delay requirement refers to the acceptable window of delay from the transmitter to the receiver. It is assumed that each user can send two types of services with different delay tolerance in the service layer. The edge layer represents the ENs with specific communication, computing and caching capabilities, which are regarded as agents. The cloud layer represents the network structure of the cloud, which including the service request list (SRL), the deployment of EN, and cloud control (CC). They have the capacity of communication, computing and storage, and can cooperate to complete cloud services.

The deployment location of the ENs determines the service they carry out. For example, the nodes deployed in the substation perform services such as transmission line monitoring and power dispatching automation, while the nodes deployed on the device side can assume the role of smart meters and perform services such as data transmission and service analysis. When the amount of data or services increase, the edge node may be limited by its own capacity and cannot meet the service delay, which violates the original intention of introducing edge computing. Therefore, it is necessary to optimize the resources of edge nodes to reduce the service delay in the case of limited resources.

By deploying ENs close to users, services will be handled by ENs first. There are K ENs within the management area and let $\mathcal{K} = \{1, 2, \ldots, K\}$ denotes the set of ENs, which are randomly distributed in the management area. In addition, the kth EN has its own cache space, computing capability, communication states, and deployment location, so the characteristic of kth EN is defined as

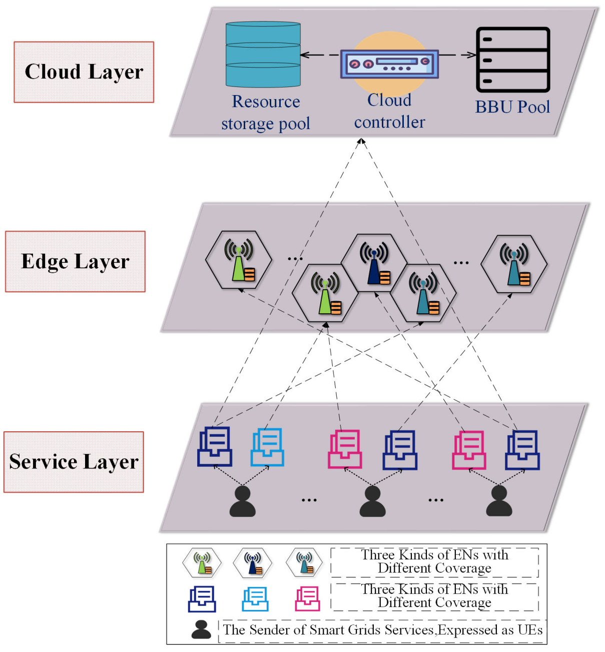
Cloud Layer
Resource storage pool
Cloud controller
BBU Pool
Edge Layer
Service Layer
...
Three Kinds of ENs with Different Coverage
Three Kinds of ENs with Different Coverage
The Sender of Smart Grids Services,Expressed as UEs

FIGURE 1. System model with multi-service and multi-EN.

$$\mathcal{C}_k = [C_k^{\text{cac}}, C_k^{\text{cop}}, C_k^{\text{com}}, C_k^{\text{pos}}]. \tag{1}$$

A. COMMUNICATION MODEL

There are U users and let $\mathcal{U} = \{1, 2, \ldots, U\}$ denotes the user set. It is assumed that different ENs share the same spectrum, the UEs associated with one EN are assigned orthogonal channels. Therefore, the interference between different ENs is taken into consideration, and there is no intra-EN interference [24]. The received signal-to-interference and noise ratio (SINR) between user u and EN k in time slot t is expressed as

$$\text{SINR}_{u,k} = \frac{g_{u,k}(t)p_{u,k}(t)}{\sigma_u^2(t) + \sum\limits_{i=1, i\neq k} g_{u,i}(t)p_i(t)}, \tag{2}$$

where $g_{u,k}(t)$ denotes the average channel gain between user u and EN k in the time t, and $g_{u,i}(t)$ is the average channel gain of communication links from other ENs. The transmission power between user u and EN k in the time t is denoted as $p_{u,k}(t)$, and $p_i(t)$ is the total transmission power for EN i in the time t. $\sigma^2(t)$ is the Gaussian white noise power in the time t.

The total bandwidth of kth EN is B_k, and the number of subchannel of each channel is M and $\mathcal{M} = \{1, 2, \ldots, M\}$, so the bandwidth allocated to each subchannel is $b_{u,k,m}$ and it represents the bandwidth between user u and EN k when subchannel m is occupied. $x_{u,k}^{\text{sub}}[m] \in \{0, 1\}$ is denoted that whether the subchannel m is allocated to the request between the user u and the EN k. With the integration of 5G and smart grids, the characteristics of some delay-sensitive services are similar to the main characteristics of URLLC, all with short packet transmission characteristics, which is to ensure ultra-low delay. In this case, the law of large numbers is considered invalid, and the system capacity cannot continue to be measured by Shannon's capacity, so the short packet and finite blocklength coding mechanisms can be used [25]–[27]. The maximum transmission rate of each communication link can be denoted as

$$r_{u,k} = \sum_{m=1}^{M} x_{u,k}^{\mathrm{sub}}[m] b_{u,k,m} \cdot [\log_2(1 + \mathrm{SINR}_{u,k}) - \sqrt{V_{u,k}/L_{u,k}} f_Q^{-1}(\varepsilon)], \tag{3}$$

where $V_{u,k} = 1 - 1/(1 + \mathrm{SINR_{u,k}})^2$ is channel dispersion and $V_{u,k} \approx 1$ in high SINR scenarios [25]. $L_{u,k}$ is the packet size. $f_Q^{-1}(\varepsilon)$ is the inverse function of the Q, and $Q(\varepsilon) = \int_{\varepsilon}^{+\infty} \frac{1}{\sqrt{2\Pi} e^{-\frac{1}{2}t^2}} dt$, where ε is error rate. If $\varepsilon \approx 0.001$, $f_Q^{-1} \approx -0.5$.

B. COMPUTATION MODEL

The computing capability of kth EN is expressed as the CPU operation cycles per second, denoted as C_k in $(cycles/sec)$, and $c_{u,k}$ denotes the computing resource allocated to UE u from the EN k. Note that delay-sensitive services may be transmitted to the cloud when ENs not have enough resources, and delay-insensitive services may be transmitted to the ENs. Therefore, for the input data from UE u to EN k generated on the smart devices, the data size is denoted as $l_{u,k}^{\mathrm{I}}$. The delay of data transmission is $d_{u,k}^{T} = l_{u,k}^{\mathrm{I}}/r_{u,k}$. Let $x_{u,k}^{\mathrm{cop}}$ be a binary variable denoting the decision for computation task from user u to the kth EN, where $x_{u,k}^{\mathrm{cop}} = 1$ means the EN k decides to execute the computation task from UE u by computing locally. The computation delay at the edge is defined as $d_{u,k}^{\mathrm{P}} = C_u/c_{u,k}$, where C_u is defined as the computing resources for completing the task from UE u.

If $x_{u,k}^{\mathrm{cop}} = 0$, it means that the computation task needs to be transmitted to the cloud as far as possible. It is assumed that the computing resources in the cloud are sufficient compared with ENs, so the total delay for processing the tasks in the cloud is regarded as constant D_u^{cop}. Comprehensively, the delay for the computation tasks can be expressed as

$$d_{u,k}^{\mathrm{cop}} = d_{u,k}^{T} + d_{u,k}^{\mathrm{P}} x_{u,k}^{\mathrm{cop}} + D_u^{\mathrm{cop}}(1 - x_{u,k}^{\mathrm{cop}}). \tag{4}$$

C. CACHE MODEL

Caching strategy can improve the communication efficiency among nodes. For example, monitoring, alarm, and control systems need to communicate with each other among multiple nodes. Since it needs a large number of signals (information) to run the monitoring and alarm system, the signal transmission has a high signaling cost. If the content based on popularity is cached in the appropriate node, it can not only reduce the signaling cost, but also improve the efficiency of information transmission.

ENs are connected to the cloud through the backhaul link, and the cache states are shared among the devices. The caching capacity of kth EN is limited, and it can store at most P popular contents, expressed as $\mathcal{P}_k = \{1, 2, \ldots, P\}$. The requested probability of content p is formulated as $q_p = \left(p^{\mathfrak{p}} \sum_{i=1}^{P} i^{-\mathfrak{p}}\right)^{-1}$, where $\mathfrak{p}_\omega = \omega_{k,p}^{\mathrm{pop}}[\alpha_k]$ is the popular factor of content p.

n_p^T represents that the number of times content p has been requested in time T, and it can be regarded as an independent and identically distributed random variable whose expectation is $E(n_{k,p}^T)$. It is assumed that the request arrival rate follows the Poisson process with an average rate λ, and the content access follows the Zipf distribution [28]. Therefore, the requests for content p can be defined as

$$\begin{aligned} E(n_{k,p}^T) &= \lambda \cdot f_{k,p}(Zipf) \\ &= \lambda \cdot (\mathfrak{p}_\omega)^{-1} \cdot \sum_{i=1}^{p} i[\alpha_k], \end{aligned} \tag{5}$$

where $f_{k,p}(Zipf)$ is distribution Zipf function, α_k is shape factor.

SMs and AMI have different importance according to their deployment location, so the centrality of device $c(k)$ is used to indicate the importance of devices. K is the number of devices in the network, $\mathcal{D}(k)$ is degree centrality, and the f is a sort function. Therefore, the formula can be described as

$$c(k) = \frac{f(\mathcal{D}(k))}{K}. \tag{6}$$

The parameter $\tau(p)$ is used to measure time threshold of each request and calculate their respective global ranking. It represents the service's tolerance for cache time. The threshold of request p is $T(p)$, and f gives a rank of all the requests, which can be given by

$$\tau(p) = \frac{f(T(p))}{P}. \tag{7}$$

The cache decision is determined by the above the centrality of devices $c(k)$ and delay threshold $\tau(p)$.

It is assumed that the cloud stores all the content and doesn't need to be updated. When the cache content of ENs reaches the maximum capacity, it needs to update the buffer pool. Considering the requirements of some services on the time of content cache, such as the intelligent monitoring of general electric equipment based on video [29], [30], a method combining popularity and cache time (PACT) is proposed to update the content. As for PACT, the content popularity represents the frequency of content being requested, and the cache time represents the length of time content that has been cached, which can be expressed as

$$\Pi^{\mathrm{cac}} = T_p^{\mathrm{cac}} \sum_{i=1}^{p} \omega_{k,p}^{\mathrm{pop}}[\alpha_k]/\omega_{k,i}^{\mathrm{pop}}[\alpha_k] \sum_{i=1}^{p} T_i^{\mathrm{cac}}, \tag{8}$$

where T_i^{cac} and T_p^{cac} are the cache time.

When the UE u requests content from EN k, if the content p is stored in the memory, it is denoted as $x_{u,k}^{\mathrm{cac}} = 1$. Otherwise, $x_{u,k}^{\mathrm{cac}} = 0$, and EN needs to get the content first, then send the content to the user. The multiplication by the content popularity is used to prefer to cache those contents with higher popularity. It is assumed that the time to obtain the content and the time to return the content to EN is a constant D^{cac}, and the cache of popular content p can be expressed as $d^{Q}_{u,k} = D^{\text{cac}}(1 - x^{\text{cac}}_{u,k})(1 - q_p)$. If the upper limit of EN cache is reached, the cache is updated according to the cache policy Π^{cac}. Therefore, the delay for the cache tasks can be given as

$$d^{\text{cac}}_{u,k} = d^{T}_{u,k} + d^{Q}_{u,k}. \tag{9}$$

D. OPTIMIZATION MODEL

The ENs make decisions for UE scheduling, computing offloading and content caching to reduce the delay of task processing. They receive service requests from users, including the type of service and SINR. ENs has computing and caching capabilities, and determines resource allocation, whether to cache content, whether to update cache, and where to perform tasks.

The location of UE u is denoted as $l_u(x, y)$ and the location of EN k is $l_k(x, y)$. The maximum coverage of EN k is denoted by $R_{u\to k}$, then a communication constraint between UE u and EN k is expressed as

$$\|l_u(x_i, y_i) - l_k(x_j, y_j)\| < R_{u\to k}. \tag{10}$$

For any service, it can only be allocated to one EN to process as follows

$$\sum_{k=1}^{K}\sum_{m=1}^{M} x^{\text{sub}}_{u,k}[m] \in \{0, 1\}, \quad \forall u. \tag{11}$$

The delay of service j must be less than a maximum tolerance value D_i, and we have

$$(d^{\text{cop}}_{u,k}(j) + d^{\text{cac}}_{u,k}(j)) < \text{D}_i, i \in \mathcal{S}^{\text{ser}}. \tag{12}$$

The allocation of bandwidth needs to be under the condition that the existing resources are sufficient, yielding the following constraint

$$\sum_{u=1}^{U}\sum_{m=1}^{M} x^{\text{sub}}_{u,k}[m] b_{u,k,m} \le B_k, \quad \forall k. \tag{13}$$

The total computing resources allocated to users by the EN k need to meet its own resource constraints, which is given by

$$\sum_{u=1}^{U} c_{u,k} \le C_k, \quad \forall k. \tag{14}$$

The optimization goal of this paper is to minimize the expected delay of N services, and the above constraints are fully considered. The optimization objective can be expressed as

$$\begin{aligned} &\arg\min_{\{d\}} \mathbb{E}[\sum_{j=1}^{N} d^{\text{cop}}_{u,k}(j) + d^{\text{cac}}_{u,k}(j)], \\ &\text{s.t.}: \quad (9) - (13). \end{aligned} \tag{15}$$

III. PROPOSED ALGORITHM

In this part, we model the resource allocation problem among multiple devices based on DRL. The goal of the algorithm is to optimize the resource scheduling strategy and reduce the service delay.

A. REINFORCEMENT LEARNING ALGORITHM

In the traditional RL algorithm, the Q-value function is solved by iterative Behrman equation, and the Q-value function is expressed as

$$Q_{i+1}(s, a) = E_s[r + \gamma \max_{a'} Q_i(s', a')|s, a]. \tag{16}$$

when $i \to \infty$, the state-action value function will eventually converge through continuous iteration, and the optimized strategy will be obtained $\pi^* = arg\max_a Q^*(s, a)$.

In this paper, the channel state, computing resources and cache state are all dynamic. When the system state changes, the size of action and state space cannot be estimated and the cost of solving Q-value function with Behrman equation is too high. With the rapid development of DQL, complex high-dimensional data can be used as input, and then deep Q-network (DQN) makes decisions according to the input data. For DRL, the space of reward is $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$, the space of Q-value is $\mathcal{S}\times\mathcal{A}$, the number of neurons is $\mathcal{X} \propto \mathcal{S}\times\mathcal{A}\times\mathcal{S}$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The complexity for solving problem (14) is expressed as $O_{DRL} = (E\cdot(\mathcal{S}\times\mathcal{A}\times\mathcal{S})\cdot T)$, where E is the number of episodes, T is the number of steps. When the number of services increases, the data can be used as the input of neural network to fit the data, which reduces the computational complexity of the traditional iterative method.

To avoid the correlation between samples, an experience replay mechanism (ERM) is introduced. The motivation of ERM is to break up the correlation within each sample, which is denoted by (s_t, a_t, r_t, s_{t+1}). s_t is the state information of the agent at time t, a_t is the action at time t, and r_t denotes the reward value under s_t and a_t. The ERM uses a limited memory to store previous experiences, and randomly selects a fixed number of experiences to update the neural network. By mixing the recent experience with the old experience to update the network, the temporal correlations existing in the replay experience is avoided [31].

B. AGENT POLLING UPDATE DEEP REINFORCEMENT LEARNING

The requirements of services in smart grids for ENs are often different. The states and actions of all agents are centrally stored by the traditional multi-agent algorithms that are conducive to global optimization, but in the face of a large number of unknown services, some ENs need special operations such as computing offloading, so the traditional method has high limitations. We use an Agent Polling Update Deep Reinforcement Learning (APUDRL) to store the agent's action separately and store the agent's state centrally. The APUDRL can increase the flexibility of the network on the premise of sharing resources among multiple ENs. For example, in a physical environment with K ENs, the service with delay tolerance of milliseconds is jointly processed by k_1 ENs, and the service with delay tolerance of seconds is jointly processed by k_2 ENs. This method can reduce the complexity of the network, reduce the waste of computing resources caused by the joint update, and make the algorithm more suitable for smart grids scenarios.

When the service needs to be processed by a single device, each EN can be regarded as an independent learner, only need to apply the APUDRL algorithm to explore the best strategy, update operation and reward. If the service needs to be processed by multiple devices, the APUDRL algorithm is used to observe its states and the states of other agents, and the operation is selected according to the joint states. When the APUDRL algorithm learns the best strategy or reaches the maximum number of times, the network can get the expected cumulative return. Markov Decision Process (MDP) is a general framework to solve DRL. By mapping optimization problems to MDP, DRL is used to solve MDP problem. The multi-service resource optimization problem in this paper is described as an MDP, which contains several key elements, including state, behavior and reward, and the details are as follows.

1) STATE

The state is the observation of the current environment, and the agent can make action decisions (see below) based on the environment. The observation includes the information of devices and the cache storage. The information of devices includes the location of kth agent $l_k(x, y)$, the computing capacity C_k^{cop}, and the state of subchannel $x_{u,k}^{\text{sub}}[m]$. The information of the cache storage includes the cached content $\varrho_{k,p}^{\text{con}}$ in the cache pool of kth agent and the popularity of the content $\omega_{k,p}^{\text{pop}}$, they are combined as the cache space C_k^{cac} of agent k, and the cache space is denoted as

$$C_k^{\text{cac}} = [[\varrho_{k,1}^{\text{con}}, \varrho_{k,2}^{\text{con}}, \dots, \varrho_{k,P}^{\text{con}}],\ [\omega_{k,1}^{\text{cac}}, \omega_{k,2}^{\text{pop}}, \dots, \omega_{k,P}^{\text{pop}}]]. \tag{17}$$

For some delay-sensitive services, the high requirements on the agent make a single agent unable to meet the service demand, so multiple agents are required to handle services cooperatively. To improve the observation information of the system, it is sensible for an agent to observe some information about the results of other involved agents. If all states information can be regarded as a vector, the states can be expressed as

$$\begin{aligned}\mathbf{s} = [&[l_1, C_1^{\text{cop}}, x_{u,1}^{\text{sub}}[m], C_1^{\text{poo}}],\\ &[l_2, C_2^{\text{cop}}, x_{u,2}^{\text{sub}}[m], C_2^{\text{poo}}], \dots,\\ &[l_K, C_K^{\text{cop}}, x_{u,K}^{\text{sub}}[m], C_K^{\text{poo}}]].\end{aligned} \tag{18}$$

2) ACTION

The action means selecting the appropriate agent to process the service. If the action of offloading to ENs or cloud is selected, the resources will be allocated to these services. If the resources of the ENs are insufficient, the ENs or the cloud can provide help. In the proposed model, the agent will select the appropriate ENs for the arriving service. According to the caching information and remaining computing resources, each agent chooses an action from the action space under the current observed state $\mathbf{s}$, and it is discrete. The action matrix can be expressed as

$$\begin{aligned}\mathbf{a} = [&a_{u,1}^{\text{com}}, a_{u,2}^{\text{com}}, \dots, a_{u,k}^{\text{com}}, \dots, a_{u,K}^{\text{com}},\\ &a_{u,1}^{\text{cop}}, a_{u,2}^{\text{cop}}, \dots, a_{u,k}^{\text{cop}}, \dots, a_{u,K}^{\text{cop}},\\ &a_{u,1}^{\text{cac}}, a_{u,2}^{\text{cac}}, \dots, a_{u,k}^{\text{cac}}, \dots, a_{u,K}^{\text{cac}}],\end{aligned} \tag{19}$$

where

$a_{u,k}^{\text{com}}$ denotes the service from UE u communicates with the EN k.

$a_{u,k}^{\text{cop}}$ denotes that the EN k will process a computation service from UE u.

$a_{u,k}^{\text{cac}}$ denotes that the EN k will process a cached service from UE u.

3) REWARD

The reward function is the reward brought by the selected action in the current state, then the agent adjusts the exploration policy according to the reward. In the APUDRL algorithm, the agent interacts with the environment to obtain the state $\mathbf{s}$, selects the action $\mathbf{a}$ according to the input state $\mathbf{s}$, and gets the real-time reward. Finally, it gets the current cumulative reward by accumulating the previous real-time reward. The objective of this paper is to reduce the service delay, so the value of the reward function has a convergent minimum. The agents learn many times to find an optimal strategy to minimize the service delay. The reward function is related to the optimization objective and it is defined as follow:

$$r_t = \alpha_1\alpha_2 d_{u,k}^{T} + \alpha_3[d_{u,k}^{\text{P}} x_{u,k}^{\text{cop}} + D_u^{\text{cop}}(1 - x_{u,k}^{\text{cop}})] + \alpha_4 d_{u,k}^{Q} - \kappa, \tag{20}$$

where

α_1,α_2,α_3 and α_4 are the penalty factors of the constraint(10), constraint(11), constraint(13) and constraint(14). When the constraint condition is not satisfied, the penalty factor is a larger value. Otherwise, the penalty factor is 1.

κ is the penalty factor of the constraint(12). It indicates whether the current policy meets the delay requirement of the service. When the constraint condition is not satisfied, the penalty factor is a larger value. Otherwise, the penalty factor is 0.

The long-term accumulative reward is defined as the sum of all rewards ever obtained by the agent, but the discounted is in each step of reward. In order to achieve the goal of minimizing task delay, the long-term cumulative reward is defined as $R_t = \sum_{t=0}^{T} \xi^t r_t$ with a discount factor $\xi \in [0, 1]$. ξ represents the impact of past rewards on current status.

4) NETWORK

The structure of APUDRL is shown in Fig. 2. The algorithm collects different services characteristics from the environment and transmits the observation values to each agent. Each agent achieves its interaction with the environment and selects actions from the action space trying to minimize

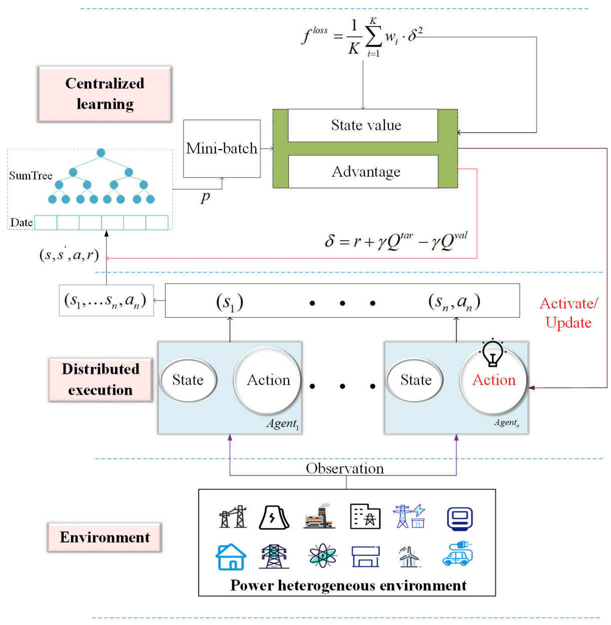

FIGURE 2. Network model of APUDRL.

the cumulative reward. An experience replay buffer $\mathcal{D}$ is used to store the training data, and a minibatch from $\mathcal{D}$ are sampled for training the network to promote development. The proposed algorithm keeps exploring until $\mathcal{D}$ reaches the upper limit, then stores the current states, actions, rewards and successor states of the agent, and they are defined by the tuple $\{\mathbf{s}, \mathbf{a}, r, \mathbf{s}'\}$. The weight of data is added to measure the standard of data update, and the tree-based SumTree is introduced to improve the sampling efficiency. The weight w of the samples is affected by the time difference (TD) error, representing the difference between the value function Q^{val} and the target value Q^{tar}, and the TD error is defined as

$$\delta = r + \gamma Q^{\text{tar}} - \gamma Q^{\text{val}}. \tag{21}$$

A high TD error indicates that the samples have a high update efficiency for the network. The principle of the experience replay mechanism is as follows. First, we use the absolute time difference TD error of experience to evaluate the learning value of experience, which has been calculated in reinforcement learning algorithms such as SARSA [32]. Then, by ranking the experiences in the replay buffer through the absolute value of their TD errors, we more frequently replay those with high magnitude of TD errors. The change of accumulated weight is defined as

$$\Delta w \leftarrow \Delta w + w^{\text{cur}}\delta \cdot \nabla Q^{\text{val}}, \tag{22}$$

where Δw is the change of accumulated weight and it needs to be reset to zero after the weight is updated and w^{cur} is the current weight value. The structure of SumTree is shown in Fig. 3 and the update function of the network is expressed as

$$f^{\text{loss}} = \frac{1}{K}\sum_{i=1}^{K} w_i \cdot \delta^2, \tag{23}$$

where f^{loss} is a residual model that represents the square of the difference between the true value and the predicted value.

The state-action value function $Q(s, a)$ of APUDRL is the result of the output of two streams in the full connection layer, namely state value function V and advantage function A. The state-action function is expressed as

$$Q(s, a) = V(s) + A'(s, a), \tag{24}$$

where $Q(s, a)$ is the parameter estimation for the network function, and the $V(s)$ provides an estimate for the state-value function. The $A'(s, a) = A(s, a) - \frac{1}{\mathbf{a}}\sum_{a'} A(s, a')$ plays an important role in advantage estimation, and it makes the network model to implement forwarding mapping to ensure that $V(s)$ and $A(s, a)$ can be recovered when $Q(s, a)$ is given.

Therefore, the APUDRL algorithm can be expressed as Algorithm 1.

Algorithm 1 Agent Polling Update Deep Reinforcement Learning Algorithm

```
1:  **Initialize** the scenario and services.
2:  **Initialize** the number of ENs $K$, the location of ENs $l$, agent attributes.
3:  **Initialize** action-value function $F^{\text{act}}$ and target action-value function $F^{\text{tar}}$, and batch size $C^{\text{bat}}$.
4:  **for** $episode = 1$ to $episode^{\max}$ **do**
5:      **Initialize** a series of random actions and receive a series of initial state $\mathbf{s}$.
6:      **for** $k = 1$ to $K$ **do**
7:          **for** $step = 1$ to $step^{\max}$ **do**
8:              Select the action and keep exploration.
9:              Acquire the action $\mathbf{a}$, reward $r$ and the next state $\mathbf{s}'$.
10:             Store $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', w)$ in the SumTree until the batch size $C^{\text{bat}}$.
11:             Get mini-batch date from the SumTree based on $p$.
12:             Activate part of the agents, and get $F^{\text{act}} = V(\mathbf{s}) + C(\mathbf{s}, \mathbf{a})$.
13:             $y_j = \begin{cases} r_j, & \text{if episode terminates at j + 1} \\ r_j + \gamma \max F^{\text{tar}}, & \text{otherwise} \end{cases}$
14:             Use the loss function $f^{\text{loss}} = \frac{1}{k}\sum_{i=1}^{K} w_i \cdot (y_j - F^{\text{act}})^2$ to update the network.
15:             Use the $\delta = y_j - F^{\text{act}}$ to update the $w$ in the SumTree.
16:             **if** the number of agents is not enough **then**
17:                 $k = k + 1$.
18:             **else**
19:                 break.
20:             **end if**
21:         **end for**
22:     **end for**
23: **end for**
```

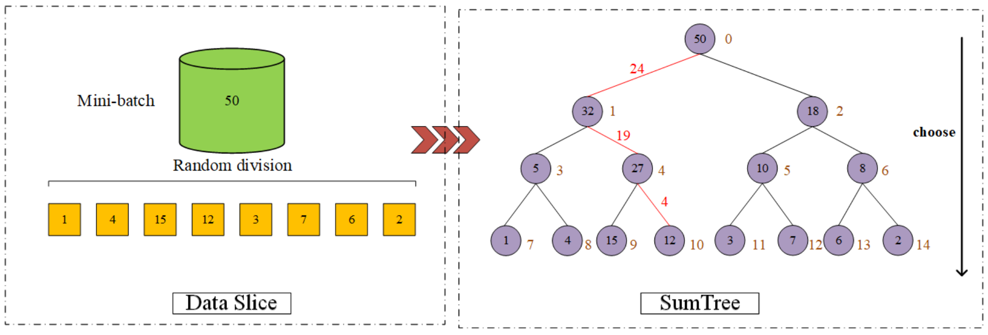

FIGURE 3. The Structure of SumTree.

TABLE 1. Hyper parameters in algorithmic networks.

Hyper Parameters	Value
Discount factor	0.9
Learning rate	0.005
E-greedy	0.9
Initial epsilon	0.01
Max-epsilon	0.9
Initial importance of sample	0.4
Growth rate of sample	0.001
Replace target iterations	200
Memory size	3000
Batch size	32
Episode number	400

IV. SIMULATION RESULTS AND ANALYSIS

In this section, we conduct extensive simulations to verify the performance of the proposed scheme. In particular, we provided simulation settings in Part A, including model parameters and network parameters. Part B demonstrates the impact of network structure and sampling method on performance. Part C analyzes the performance of the resource allocation strategy from two aspects of computing offload and cache hit rate.

A. SIMULATION SETTINGS

The important parameters in performance evaluation are listed in Table 1, and the specific parameters are analyzed as follows.

The system environment of a single macro cell with a radius of 5km is considered. It is assumed that the maximum communication distance of ENs is 100m [5]. The ENs are randomly located in the system environment according to a uniform distribution. The wireless transmission capacity (WiFi) and wired transmission capacity (power line communication) are set to 2.5Mbps and 15Mbps respectively. The maximum amount of cache capacity is $C^{cac} = 20$ and the request list capacity is $C^{list} = 100$. The probability of computing task generation is 0.6, and cache task generation is 0.4. The distribution of content popularity is often Zipf distribution, and the shape factor of the is 0.56 [33].

The initial value of the importance of the sample is 0.4, and its growth rate is 0.001. The agent is composed of two neural networks, namely the target network and the evaluation network. The target network keeps random exploration and trains the network. The evaluation network evaluates the results of the target network. The parameters of both networks are replaced with the rate $v^{rep} = 0.001$ once every 200 iterations. For training the neural network, the size of the experience replay buffer is 3000, and it returns a mini-batch of experiences, the size of the mini-batch is 32. The learning rates of several algorithms all are set as $v^{lea} = 0.01$.

B. TRAINING PERFORMANCE EVALUATION

In this part, we focus on the network performance of the algorithm and compare the performance of the proposed algorithm with that of three baseline algorithms, namely Double DQN [34], Prioritized Replay DQN [35] and Dueling DQN [36]. The algorithm converges approximately after several iterations, and the performance is evaluated by reward function and loss function. For these three baseline algorithms, their network structures and sampling methods are different, but they are usually used to deal with the resource allocation problem of discrete actions.

1) DOUBLE DQN

Double DQN has the same structure of two Q-networks. Through decoupled the choice of target Q-value action and the calculation of target Q-value, it can resolve the over-estimation of DQN.

2) PRIORITIZED REPLAY DQN

Prioritized replay DQN with SumTree uses priority sampling to improve prediction accuracy.

3) DUELING DQN

The Dueling DQN tries to optimize the algorithm by optimizing the structure of the neural network. By dividing the Q-network into two parts, one is the state value function $V(s)$ and the other is the advantage function $A(s, a)$.

Fig. 4 shows the total reward per episode of the proposed algorithm and three other algorithms in the same environment. The total rewards per episode of all algorithms first decrease with the increase of the episode, and then due to the influence of random learning strategy, the rewards have some small fluctuations, but finally converges. It can be seen that the gain of the APUDRL algorithm in terms of

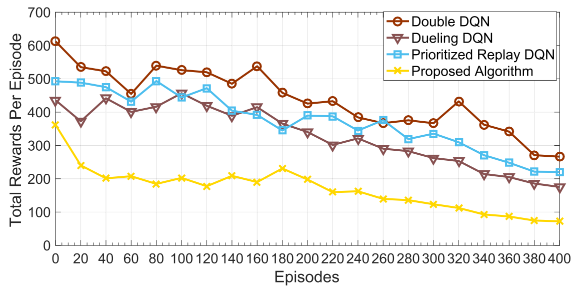

FIGURE 4. Total rewards achieved by different algorithms.

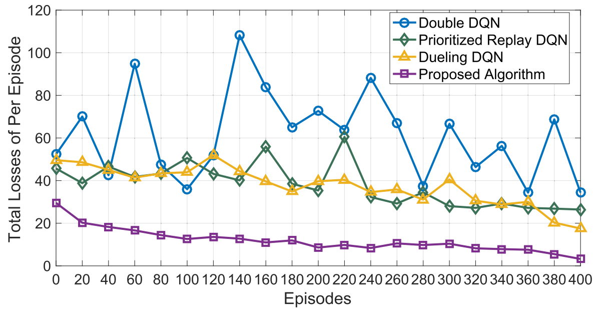

FIGURE 5. Total losses achieved by different algorithms.

reward value is approximately 72.85%, 61.65% and 58.84% compared with Double DQN, Prioritized Replay DQN and Dueling DQN. The convergence value under the Double DQN algorithm reaches 266.35 which has the worst performance, and the best performance is 72.24 from the proposed algorithm. The reasons are as follows. First, the two algorithms rely on different neural network structure. Specifically, the neural network of Double DQN uses a uniform sampling method which has poor adaptability to newly taken actions, resulting in extremely unsatisfactory returns from the algorithm. In contrast, the neural network of the proposed algorithm introduces SumTree to optimize data storage, and employs an advantage function to evaluate currently taken actions. Furthermore, the sampling method of the algorithm is efficient, and the advantage function improves the accuracy of decision-making. For the newly added actions, the network can learn quickly and estimate the reward in advance.

In Fig. 5, the loss function of the algorithms decreases with the increasing episode. For Double DQN, the fluctuate is unsatisfactory and it does not reach the convergence value in the end, because the sampling method and network structure lead to inaccurate strategy. As can be seen in the figure, the proposed algorithm can obtain the total losses per episode fluctuates around 5 and it gives a lower bound to the other algorithms. As expected, the fluctuation range and convergence value of the loss function of the proposed algorithm significantly outperform the other three algorithms since the agents in proposed algorithm have efficient sampling methods and accurate decision-making.

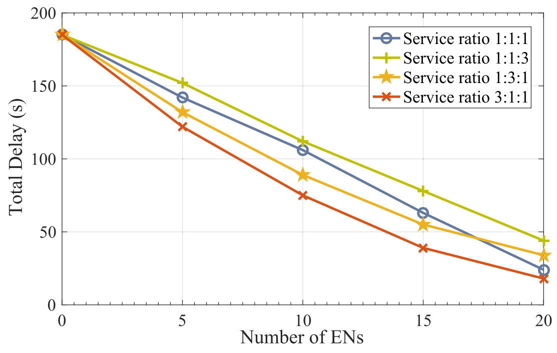

FIGURE 6. Cache capacity versus cache hit ratio under the three cache update policy.

C. PERFORMANCE OF COMPUTING AND CACHING

In this part, we evaluate the performance of computing and caching based on the proposed algorithm. Considering the different requirements of service delay in smart grids, we analyze the service delay of three kinds of services in different proportions. The ratio of milliseconds and seconds to minutes is 1:1:1, 1:1:3, 1:3:1 and 3:1:1 respectively. As shown in Fig. 6, the number of ENs is set from 0 to 20. When the network resources are sufficient, the total processing delay approximately presents a linear downward trend with the increase of the number of ENs. The delay rapidly decreases with the increasing number of milliseconds services and delay slowly decreases with the increasing number of minutes services and seconds services. The results show that the proposed algorithm can adapt to the service with different delays, and change the strategy adaptively according to the service delay requirements to ensure the convergence of the results.

Fig. 7 shows the total delay of the proposed algorithm with different number and proportion of services, the three different computing offloading modes including [37]

- ENs Computing: All services are delivered to ENs for processing. This is suitable for the situation that the service load is light and the ENs have abundant resources to finish the task in time.
- Cloud Computing(CC): All services are delivered to CC for processing. Compared with ENs computing, CC has more resources and is suitable for services with lower latency requirements because this transmission costs more resources and time.
- ENs and CC: All services can choose to process in ENs or deliver to the cloud for processing, which is suitable for a variety of services with different delay requirements.

As shown in Fig. 7, the delay of the single service is always lower than that of mixed service the number of services. The single service refers to that the service requested by the user is composed of one type of service in $\mathcal{S}^{\text{ser}}$, while the mixed service refers to that the service requested by the user is composed of multiple types in $\mathcal{S}^{\text{ser}}$. When the number of

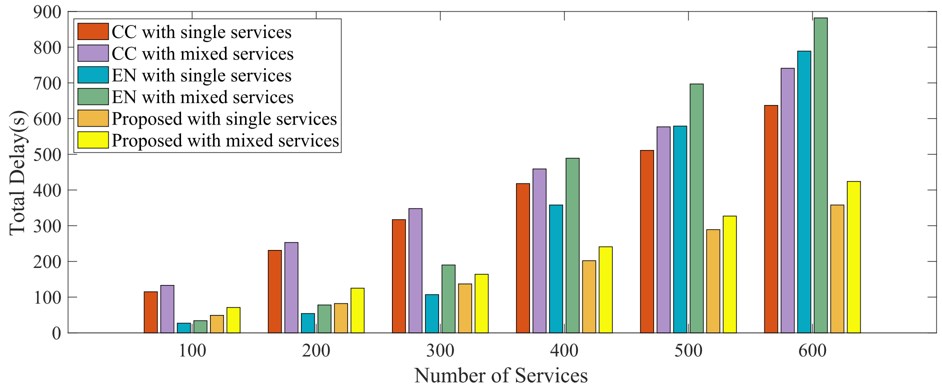

FIGURE 7. System delay versus the number of services with different computing offloading and service ratios.

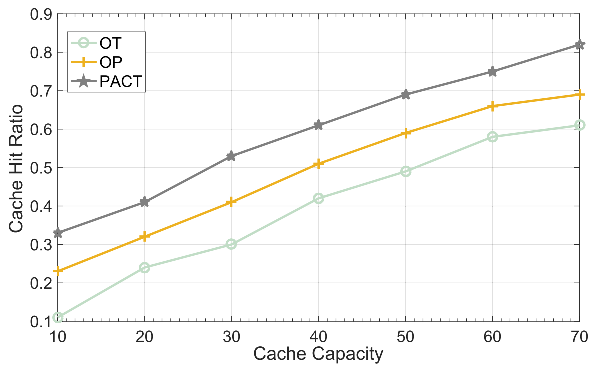

FIGURE 8. Cache hit rate of OT, OP and PACT caching schemes for different cache capacity.

services is less than 300 and the proportion of services is the same, the delay of all services delivered to ENs is lower than that of the other two. This is because the resources of ENs can handle a small number of services. However, with the increase in the number of services, this method is limited due to resource constraints. On the whole, the combination of ENs and CC in this paper can adapt to the calculation of multiple types of services under different traffic volumes. It not only ensures that the ENs has enough resources to deal with delay-sensitive services, but also makes effective use of the characteristics of CC, alleviating the dual computing pressure of ENs and CC.

Fig. 8 shows the comparison of cache hit rates among OT, OP, and PACT, where OT only considers cache time and the OP only considers content popularity. As can be seen from Fig. 8, the OT has the lowest cache hit rate, which is always lower than the other two strategies. It is illustrated that the OT can only adapt to the caching requirements of a few services, such as power monitoring video. The cache hit rate of PACT is always higher than that of OT and OP. The obtained shows that considering the content popularity and content cache time, the cache hit rate has a significant improvement under different cache capacity. Compared with OT and OP, the cache hit rate of the PACT is improved by 34.42% and 18.71%.

V. CONCLUSION

We design an EC system framework with three-layer in smart grids combining EC and CC to allocate computing and caching resources, which is suitable for a large number of services in smart grids with different delay requirements. A DRL algorithm based on a polling method that adapts to the smart grids is proposed, which allows the agent to perform the polling mechanism according to the requirements of the service to optimize the neural network and the constraints in optimization problems are mapped into penalty factors of reward functions. Numerical experiments show that, compared with the baseline algorithm, the proposed algorithm can achieve superior long-term utility performance, which is reflected in the smaller convergence value of the reward function and loss function. In the face of the growing number of services with different delay requirements, the algorithm is also surpassed that of the baseline schemes in delay and cache hit rate.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.
[2] J.-P. Carvallo, J. Taneja, D. Callaway, and D. M. Kammen, "Distributed resources shift paradigms on power system design, planning, and operation: An application of the GAP model," *Proc. IEEE*, vol. 107, no. 9, pp. 1906–1922, Sep. 2019.
[3] G. Mei, N. Xu, J. Qin, B. Wang, and P. Qi, "A survey of Internet of Things (IoT) for geohazard prevention: Applications, technologies, and challenges," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4371–4386, May 2020.
[4] J. Zhao, C. Wan, Z. Xu, and J. Wang, "Risk-based day-ahead scheduling of electric vehicle aggregator using information gap decision theory," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1609–1618, Jul. 2017.
[5] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Comput. Netw.*, vol. 67, pp. 74–88, Jul. 2014.
[6] B. Yang, K. V. Katsaros, W. K. Chai, and G. Pavlou, "Cost-efficient low latency communication infrastructure for synchrophasor applications in smart grids," *IEEE Syst. J.*, vol. 12, no. 1, pp. 948–958, Mar. 2018.

[7] Y. Li, X. Cheng, Y. Cao, D. Wang, and L. Yang, "Smart choice for the smart grid: Narrowband Internet of Things (NB-IoT)," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1505–1515, Jun. 2018.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[9] H. Mei, K. Wang, and K. Yang, "Multi-layer cloud-RAN with cooperative resource allocations for low-latency computing and communication services," *IEEE Access*, vol. 5, pp. 19023–19032, 2017.

[10] X. Zhang, J. Wu, W. Shi, Y. Wu, and Y. Miu, "Low-latency cooperative computation offloading for mobile edge computing," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. With Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2019, pp. 155–159.

[11] J. Yao, Z. Li, Y. Li, J. Bai, J. Wang, and P. Lin, "Cost-efficient tasks scheduling for smart grid communication network with edge computing system," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 272–277.

[12] Z. Li, Y. Liu, R. Xin, L. Gao, X. Ding, and Y. Hu, "A dynamic game model for resource allocation in fog computing for ubiquitous smart grid," in *Proc. 28th Wireless Opt. Commun. Conf. (WOCC)*, May 2019, pp. 1–5.

[13] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[14] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[15] N. Saputro and K. Akkaya, "Investigation of smart meter data reporting strategies for optimized performance in smart grid AMI networks," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 894–904, Aug. 2017.

[16] X. Liu, S. Zhang, X. Zeng, L. Yao, Y. Ding, and C. Deng, "Evaluating the network communication delay with wams for multi-energy complementary systems," *CSEE J. Power Energy Syst.*, vol. 6, no. 2, pp. 402–409, 2020.

[17] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE J. Power Energy Syst.*, vol. 4, no. 3, pp. 362–370, Sep. 2018.

[18] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.

[19] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, Nov. 2019.

[20] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.* Cham, Switzerland: Springer, 2017, pp. 66–83.

[21] W. Jiang, G. Feng, S. Qin, and Y.-C. Liang, "Learning-based cooperative content caching policy for mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[22] K. K. Nguyen, T. Q. Duong, N. A. Vien, N.-A. Le-Khac, and M.-N. Nguyen, "Non-cooperative energy efficient power allocation game in D2D communication: A multi-agent deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 100480–100490, 2019.

[23] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE J. Power Energy Syst.*, vol. 6, no. 1, pp. 213–225, 2020.

[24] H. Ke, J. Wang, H. Wang, and Y. Ge, "Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 179349–179363, 2019.

[25] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[26] G. Durisi, T. Koch, and P. Popovski, "Toward massive, ultrareliable, and low-latency wireless communication with short packets," *Proc. IEEE*, vol. 104, no. 9, pp. 1711–1726, Sep. 2016.

[27] C. She and C. Yang, "Available range of different transmission modes for ultra-reliable and low-latency communications," in *Proc. IEEE 85th Veh. Technol. Conf. (VTC Spring)*, Jun. 2017, pp. 1–5.

[28] B. Chen and C. Yang, "Caching policy optimization for D2D communications by learning user preference," in *Proc. IEEE 85th Veh. Technol. Conf. (VTC Spring)*, Jun. 2017, pp. 1–6.

[29] T. Liu, "Development and application of electrical equipment intelligent detection management system," in *Proc. 5th Int. Conf. Smart Grid Electr. Automat. (ICSGEA)*, Jun. 2020, pp. 70–73.

[30] S. Hu, X. Chen, W. Ni, X. Wang, and E. Hossain, "Modeling and analysis of energy harvesting and smart grid-powered wireless communication networks: A contemporary survey," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 461–496, Jun. 2020.

[31] F. Bu and D. E. Chang, "Double prioritized state recycled experience replay," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Nov. 2020, pp. 1–6.

[32] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.

[33] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[34] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[35] X. Cao, H. Wan, Y. Lin, and S. Han, "High-value prioritized experience replay for off-policy reinforcement learning," in *Proc. IEEE 31st Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2019, pp. 1510–1514.

[36] Z. Liu, X. Chen, Y. Chen, and Z. Li, "Deep reinforcement learning based dynamic resource allocation in 5G ultra-dense networks," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2019, pp. 168–174.

[37] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2565–2578, Apr. 2021.

LINHAN XI received the B.S. degree in communication engineering from Jilin University, China, in 2019. He is currently pursuing the master's degree in telecommunication and information systems with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include smart grid, machine learning, and wireless resource management in future wireless networks.

YING WANG (Member, IEEE) received the Ph.D. degree in circuits and systems from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2003. In 2004, she was invited to work as a Visiting Researcher with the Communications Research Laboratory (renamed NiCT, since 2004), Yokosuka, Japan. In 2005, she was a Research Associate with The University of Hong Kong, Hong Kong. She is currently a Professor with BUPT and the Director of the Radio Resource Management Laboratory, Wireless Technology Innovation Institute, BUPT. She is also active in standardization activities of 3GPP and ITU. She took part in performance evaluation work of the Chinese Evaluation Group, as a Representative of BUPT. She was also selected in the New Star Program of the Beijing Science and Technology Committee, in 2007, and the New Century Excellent Talents in University, Ministry of Education, in 2009. She has authored over 100 articles in international journals and conferences proceedings. Her research interests include cooperative and cognitive systems, radio resource management, and mobility management in 5G systems. She was a recipient of first prizes of the Scientific and Technological Progress Award by the China Institute of Communications, in 2006 and 2009, respectively, and a Second Prize of the National Scientific and Technological Progress Award, in 2008.

YANG WANG received the Ph.D. degree in power system and automation from China Electric Power Research Institute Company Ltd (CEPRI), in 2015. He is currently a Senior Engineer and the Chief Engineer of the Institute of Information and Communication, Chinese Academy of Sciences. He presided over or participated in three national projects, 17 science and technology projects of State Grid, compiled two technical monographs, published 17 EI retrieval papers, and published more than 40 articles in major core journals. He has obtained 26 authorized patents, applied for six software copyrights, and compiled more than ten industry and enterprise standards.

ZHIHUI WANG received the M.S. degree in measurement technology and instrument from Beihang University, in 2006. He is currently a Senior Engineer engaged in the research and test of power communication technology for a long time. As the Project Leader or the Executive Director, he has undertaken more than ten major national and company science and technology projects, published more than 20 articles in core and above journals, and applied for more than ten invention patents. He has led the compilation of more than ten industry and enterprise standards, issued more than 1000 test reports, outstanding expert reserve talents of State Grid Corporation of China, and excellent expert talents of China Electric Power Academy.

XUE WANG received the B.S. degree in communication engineering from Jilin University, China, in 2018. She is currently pursuing the Ph.D. degree in information and communication engineering with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include smart grid, network slicing, and wireless resource management in future wireless networks.

YUANBIN CHEN received the B.S. degree in communications engineering from Beijing Jiaotong University, Beijing, China, in 2019. He is currently pursuing the master's degree in information and communication systems with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His current research interests include reconfigurable intelligent surface (RIS), vehicular networks, and radio resource management (RRM) in future wireless networks. He was a recipient of the National Scholarship, in 2020.