Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051530

ARTICLE

Cloud-Edge Collaborative Federated GAN Based Data Processing for IoT-Empowered Multi-Flow Integrated Energy Aggregation Dispatch

Zhan Shi*

Electric Power Dispatching Control Center, Guangdong Power Grid Co., Ltd., Guangzhou, 510030, China

*Corresponding Author: Zhan Shi. Email: w_1234567892021@163.com

Received: 07 March 2024 Accepted: 27 May 2024 Published: 18 July 2024

ABSTRACT

The convergence of Internet of Things (IoT), 5G, and cloud collaboration offers tailored solutions to the rigorous demands of multi-flow integrated energy aggregation dispatch data processing. While generative adversarial networks (GANs) are instrumental in resource scheduling, their application in this domain is impeded by challenges such as convergence speed, inferior optimality searching capability, and the inability to learn from failed decision making feedbacks. Therefore, a cloud-edge collaborative federated GAN-based communication and computing resource scheduling algorithm with long-term constraint violation sensitiveness is proposed to address these challenges. The proposed algorithm facilitates real-time, energy-efficient data processing by optimizing transmission power control, data migration, and computing resource allocation. It employs federated learning for global parameter aggregation to enhance GAN parameter updating and dynamically adjusts GAN learning rates and global aggregation weights based on energy consumption constraint violations. Simulation results indicate that the proposed algorithm effectively reduces data processing latency, energy consumption, and convergence time.

KEYWORDS

IoT; federated learning; generative adversarial network; data processing; multi-flow integration; energy aggregation dispatch

1 Introduction

The rapid growth of renewable energy sources such as distributed photovoltaics (PVs) and wind power brings new challenges to the energy demand-supply balance due to their volatile, random, and intermittent characteristics [1,2]. This spurs the development of novel energy dispatch services where distributed PVs, electric vehicles, and adjustable loads are intelligently aggregated and dispatched based on real-time processing of grid state data gathered by Internet of Things (IoT) devices [3,4]. Therefore, the influx of massive state data, the increasing randomness and volatility of energy sources, as well as the rising complexity of energy dispatch services, present an urgent requirement to integrate information flow, energy flow, and service flow [5]. However, the traditional cloud-based paradigm is no longer suitable to meet the exponentially growing data processing demands of multi-flow integration [6,7]. How to realize real-time and energy-efficient data processing for multi-flow integrated energy aggregation dispatch remains an open issue.

Edge computing can provide proximate data processing. It can be combined with cloud computing in a complementary fashion to realize cloud-edge collaboration and support complex data processing [8,9]. In addition, with the advancement of communication technologies such as power line communication (PLC) and 5G, cloud-edge collaboration can further improve workload balance via edge-edge migration and cloud-edge migration [10,11]. For example, data can be intelligently migrated from a heavy-loaded edge server to a light-loaded edge server to reduce processing delay and improve computing resource utilization efficiency [12,13]. The core of edge-cloud collaboration for multi-flow integrated energy aggregation dispatch lies in resource scheduling optimization. Service data migration, computing resource allocation, and transmission power control should be adaptively optimized with time-varying channel states, electromagnetic interference, server workload, and service requirements.

Generative adversarial network (GAN) is a deep learning model, which can enhance the training process by min-max zero-sum game of generator and discriminator networks to achieve Nash equilibrium. In [14], Ali et al. employed GAN to obtain the resource scheduling strategy for high-reliable low-delay communication in wireless networks. In [15], Rohit et al. adopted a GAN-assisted network slicing resource orchestration algorithm in industrial IoT applications. In [16], Hua et al. presented a GAN-assisted distributed deep learning to solve the resource scheduling problem among multiple network slices. In [17], Naeem et al. leveraged GAN-based deep distributional Q-network for learning the action-value distribution for intelligent transmission scheduling to achieve ultra-reliable low latency communication. The application of GAN in resource scheduling helps improve model robustness, optimize scheduling strategies, and reduce experimental costs, thereby facilitating the effective operation and management of power systems. However, when applying GAN to multi-flow integrated energy aggregation dispatch, several technical challenges remain to be addressed.

First, the realization of multi-flow integrated energy aggregation dispatch is indispensable from real-time and energy-efficient data processing. However, low processing energy consumption and delay are paradoxical goals. Increasing transmission power can reduce transmission delay and the total data processing delay, but reduces the future energy budget and increases the probability of energy consumption constraint violation [5]. Second, traditional GAN suffers from slow convergence speed and inferior optimality searching capability due to the lack of global foresight of workload distribution. It is intuitive to explore cloud-edge collaboration to provide a priori knowledge of the entire network for improving GAN without significantly increasing communication overheads. Last but not least, traditional GAN cannot learn from failed decision making feedbacks such as the violation occurrences of energy consumption constraint. How to augment GAN with failure occurrence sensitiveness to further improve convergence and optimality searching performances is a key challenge.

Federated learning adopts a semi-distributed learning framework. Compared to traditional centralized learning, it reduces communication costs, enhances global performance, and speeds up the model convergence. By integrating federated learning, IoT, and cloud-edge collaboration, part of the model training can be done on edge servers, reducing communication overhead between cloud and edge servers, and thus improving real-time performance [18]. Furthermore, federated learning improves the global network insight performance of GAN in a distributed environment, boosting the convergence rate. In [19], Xu et al. proposed a federated generative adversarial network (FGAN)-based decentralized data synthesizing and data processing integration method, which is able to improve the traffic classification performance. In [20], Eisuke et al. proposed an FGAN-based image generation model training method in wireless ad hoc collaboration, which achieves better performance in cross-node data learning and image generation. In [21], Sui et al. augmented the training dataset by

leveraging the FGAN method and addressed the unlabeled data by means of an active learning method. In [22], Li et al. proposed an alternative approach which learns a globally shared GAN model by aggregating locally trained generators' updates with maximum mean discrepancy to achieve the highest inception score and produce high-quality instances. However, there are still some unresolved issues in the literature mentioned above. Firstly, these works did not take into account the multi-flow integration based on IoT and cloud-edge collaboration, and did not enable joint optimization of energy consumption and delay. Secondly, edge-edge as well as edge-cloud migration are not considered, making it difficult to achieve load balancing and reduce processing delay. Lastly, the insights provided by failure events have not been effectively utilized, resulting in poor accuracy and convergence speed.

To deal with the aforementioned challenges, a cloud-edge collaborative FGAN-based data processing algorithm is proposed for multi-flow integrated energy aggregation dispatch, with the optimization objective of minimizing the long-term average weighted sum of total energy consumption and total delay under long-term energy consumption constraint. First, we develop the system models of cloud-edge collaborative migration, data processing, and total energy consumption and delay for multi-flow integrated energy aggregation dispatch. Second, we formulate a communication and computing resource scheduling optimization problem. Then, we utilize Lyapunov optimization to perform problem decomposition. Finally, a cloud-edge collaborative FGAN-based communication and computing resource scheduling algorithm with long-term constraint violation sensitiveness is presented, which effectively exploits global environment information to optimize the cloud-edge resource scheduling strategy, and achieves a joint guarantee of transmission delay and energy consumption.

The main innovations are introduced as follows:

- **Real-time and energy-efficient data processing for multi-flow integrated energy aggregation dispatch:** We formulate a weighted sum of total energy consumption and delay minimization problem for real-time and energy-efficient data processing, where the weight factor is utilized to balance the energy consumption and delay. In addition, the energy deficit virtual queue is incorporated into the optimization objective through problem decomposition, which enforces low-energy consumption resource scheduling optimization.
- **Cloud-edge collaborative FGAN-based resource scheduling algorithm:** A cloud-edge collaboration FGAN-based communication and computing resource scheduling algorithm is proposed. The edge-edge and edge-cloud data migration are considered to reduce processing delay. In addition, the cloud and edge servers cooperate to perform network training and parameter interaction to jointly optimize communication resource scheduling, such as transmission power, as well as computing resource scheduling, including data migration and computing resource allocation.
- **Improved convergence and optimality for FGAN with constraint violation sensitiveness:** We integrate federated learning with GAN to leverage global insight for improving convergence and optimality performance of resource scheduling in multi-flow integrated energy aggregation dispatch data processing. In addition, the number of constraint violation occurrences is utilized for dynamic adjustment of the learning rate and network parameter weight to achieve constraint violation sensitiveness and improve accuracy and convergence speed.

The remaining part is arranged as shown below. Section 2 presents the system model. Section 3 expatiates problem formulation and decomposition. The cloud-edge collaborative FGAN-based resource scheduling algorithm is proposed in Section 4. Sections 5 and 6 give the simulation results and conclusion.

2 System Model

The multi-flow integrated energy aggregation dispatch framework based on cloud-edge collaboration is illustrated in Fig. 1, including four layers of device, edge, cloud, and application. There is various electrical equipment in the device layer, including distributed PV, electric vehicle, charging pile, and adjustable load. Numerous IoT devices are arranged to gather key equipment state data, which are uploaded to the edge layer based on PLC. The edge layer deploys several edge servers that process the data uploaded from devices within their coverage. To reduce data processing delay and improve load balance, an edge server can also migrate its data to other neighbor servers or the cloud server for processing via edge-edge migration or edge-cloud migration. The cloud layer consists of a cloud server with large computing capacity but located far away from the device layer. Thus, edge-cloud migration reduces processing delay at the cost of increased transmission delay. The application layer operates novel services of PV dispatch, electric vehicle dispatch, load dispatch, and virtual power plant dispatch based on the data processed by edge and cloud servers.

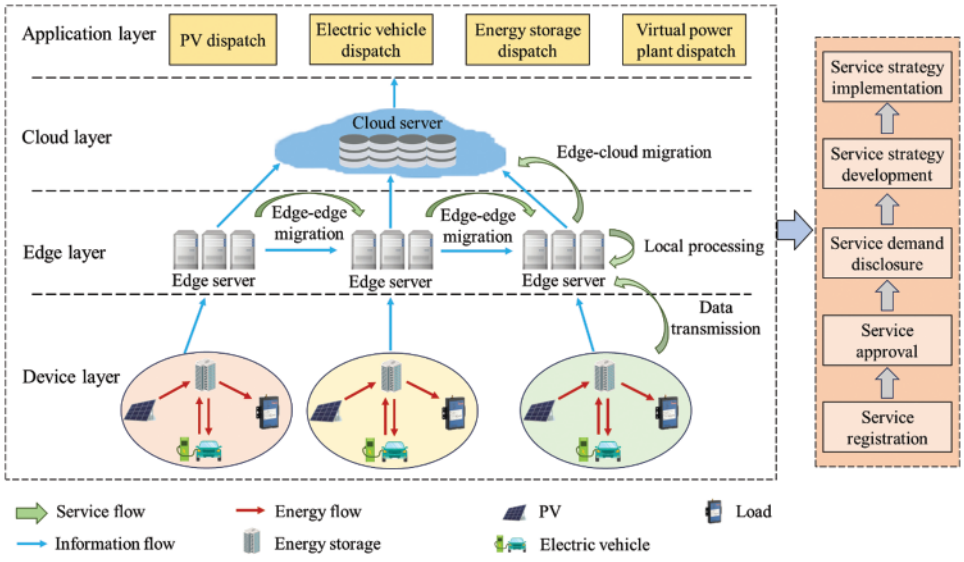


Figure 1: Multi-flow integrated energy aggregation dispatch framework based on cloud-edge collaboration

There exist information, energy and service flows based on the interaction among device, edge, cloud and application layers. Information flow contains the entire lifecycle of data collection, transmission, processing. Energy flow represents the whole process of energy generation, transmission, conversion, and utilization in energy aggregation dispatch of low-voltage distribution gird. Particularly, distributed PV generators, electric vehicles, loads, and energy storage units are intelligently aggregated and scheduled to meet the demand-supply balance. Both information flow and energy flow serve are the key pillars to realize the service flow of various applications including service registration, approval, demand disclosure, service strategy development, and implementation.

Define the set of M edge servers as $\mathcal{S} = \{s_1, \ldots, s_m, \ldots, s_M\}$. The cloud server is represented as s_{M+1}. Each edge server manages N IoT devices, and the set of devices managed by s_m is $\mathcal{D}_m = \{d_{m,1}, \ldots, d_{m,n}, \ldots, d_{m,N}\}$. A time slot represents the total duration experienced by a data packet from generation to processing. We consider a total of T time slots, i.e., $\mathcal{T} = \{1, \ldots, t, \ldots, T\}$.

Due to the time varying computing resources and workloads, edge servers are categorized into two types, i.e., heavy-loaded servers and light-loaded servers. A heavy-loaded edge server with little computing resources and overwhelming workloads will result in large processing delay. Therefore, it is intuitive to avoid heavy-loaded edge servers by optimizing edge-edge and edge-cloud migrations. Define $x_{n,m,m'}(t) \in \{0,1\}$ as the service migration indicator variable. $x_{n,m,m'}(t) = 1$ indicates that data packet is migrated. Specifically, when $x_{n,m,m'}(t) = 1$ and $m' = m$, it indicates that the data packet is processed locally by the edge server s_m. When $x_{n,m,m'}(t) = 1$, $m' \neq m$ and $s'_m \in \mathcal{S}$, it indicates that the data packet is migrated from s_m to edge server s'_m based on edge-edge migration. In addition, if $m' = M + 1$, it indicates that the packet is migrated from s_m to the cloud server based on edge-cloud migration.

2.1 Cloud-Edge Collaborative Migration Model

2.1.1 Device-Edge Data Transmission Model

At each time slot, IoT device uploads the collected data to edge layer through PLC. In slot t, the transmission delay from $d_{m,n}$ to s_m can be expressed as

$$\tau^{up}_{n,m}(t) = \frac{a_{n,m}(t)}{R^{up}_{n,m}(t)}, \tag{1}$$

where $a_{n,m}(t)$ denotes the collected data size. $R^{up}_{n,m}(t)$ is the transmission rate from $d_{m,n}$ to s_m, which is calculated as

$$R^{up}_{n,m}(t) = B^{up}_{n,m}(t) \log_2\left(1 + \frac{P^{tran}_{n,m}(t)\, g_{n,m}(t)}{\delta^2 + e_{n,m}(t)}\right) \tag{2}$$

$e_{m,n}(t)$ is the electromagnetic interference (EMI) power. $g_{m,n}(t)$ and δ^2 are the channel gain, white noise power. $P^{tran}_{n,m}(t)$ is the device-edge transmission power. $B^{up}_{n,m}(t)$ is the device-edge transmission bandwidth. $P^{tran}_{n,m}(t)$ is discretized into K levels, i.e.,

$$P^{tran}_{n,m}(t) \in \mathcal{P}^{tran} = \left\{P^{tran}_{n,m,min}, \ldots, P^{tran}_{n,m,min} + \frac{(k-1)\left(P^{tran}_{n,m,max} - P^{tran}_{n,m,min}\right)}{K-1}, \ldots, P^{tran}_{n,m,max}\right\}, \tag{3}$$

where $P^{tran}_{n,m,min}(t)$ and $P^{tran}_{n,m,max}(t)$ are the minimum and maximum transmission power of $d_{m,n}$, respectively.

To intuitively illustrate the effect of EMI on PLC, the alpha-stable distribution is employed to describe the EMI with distinct impulse characteristics [23]. Define $W\left(e^{\chi q e_{n,m}(t)}\right)$ as the eigenfunction of $e_{m,n}(t)$, whose expression is given by

$$W\left(e^{\chi q e_{n,m}(t)}\right) = \begin{cases} \exp\left(\chi \mu_{n,m}\varphi - \xi_{n,m}|\varphi|^{a_{n,m}} \times \left(1 - \chi\beta_{n,m}\mathrm{sgn}(\varphi)\tan\frac{\alpha_{n,m}\pi}{2}\right)\right), \alpha_{n,m} \neq 1 \\ \exp\left(\chi \mu_{n,m}\varphi - \xi_{n,m}|\varphi| - \chi\beta_{n,m}\mathrm{sgn}(\varphi)\ln|\varphi|\frac{2}{\pi}\right), \alpha_{n,m} = 1, \end{cases} \tag{4}$$

where $\alpha_{n,m}$, $\beta_{n,m}$, $\xi_{n,m}$, and $\mu_{n,m}$ refer to the eigenfactor, skew parameter, scale parameter and position parameter, respectively. χ and φ denote the quantization coefficients of the scale and position parameters, respectively.

The transmission energy consumption of $d_{m,n}$ is calculated as

$$E^{up}_{n,m}(t) = \tau^{up}_{n,m}(t)\, P^{tran}_{n,m}(t). \tag{5}$$

2.1.2 Edge-Edge and Edge-Cloud Data Migration Models

The data packet is migrated to the edge server or cloud server based on 5G communications. The edge-edge and edge-cloud data migration delays are calculated as

$$\tau_{n,m}^{ee}(t) = \sum_{s_{m'} \in \mathcal{S}} x_{n,m,m'}(t)\, \tau_{m,m'}^{ee}(t), m' \neq m, \tag{6}$$

$$\tau_{n,m}^{ec}(t) = x_{n,m,m'}(t)\, \tau_{m,m'}^{ec}(t), m' = M+1, \tag{7}$$

where $\tau_{m,m'}^{ee}(t)$ and $\tau_{m,m'}^{ec}(t)$ are the data transmission delay between edge sever s_m and s_m' as well as s_m and the cloud server.

The energy consumptions of edge-edge and edge-cloud data migration are calculated as

$$E_{n,m}^{ee}(t) = \sum_{s_{n'} \in \mathcal{S}} x_{n,m,m'}(t)\, E_{m,m'}^{ee}(t), m' \neq m, \tag{8}$$

$$E_{n,m}^{ec}(t) = x_{n,m,m'}(t)\, E_{m,m'}^{ec}(t), m' = M+1. \tag{9}$$

$E_{m,m'}^{ee}(t)$ and $E_{m,m'}^{ec}(t)$ are the transmission energy consumption between edge sever s_m and s_m' as well as s_m and the cloud server, and $E_{m,m'}^{ee}(t) < E_{m,m'}^{ec}(t)$.

2.2 Data Processing Model

2.2.1 Edge Data Processing

The edge data processing delay of data from device $d_{m,n}$ at edge server s_m is calculated as

$$\tau_{n,m}^{pro}(t) = \sum_{s_{m'} \in \mathcal{S}} x_{n,m,m'}(t)\, \frac{a_{n,m}(t)\, \gamma_{n,m}(t)}{\chi_{n,m,m'}^{ee}(t)}, \tag{10}$$

where $\gamma_{n,m}(t)$ is the computation complexity of processing single bit data of $d_{m,n}$. When $m' = m$, $\chi_{n,m,m'}^{ee}(t)$ indicates the computing resources allocated by s_m and the data of device $d_{m,n}$ are processed locally. When $m' \neq m$, the data are migrated from s_m to s_m' and processed remotely. $\chi_{n,m,m'}^{ee}(t)$ indicates the computing resources allocated by s_m' for $d_{m,n}$.

The edge data processing energy consumption is given by

$$E_{n,m}^{pro}(t) = \sum_{s_{m'} \in \mathcal{S}} x_{n,m,m'}(t)\, e_{n,m}^{e} a_{n,m}(t)\, \gamma_{n,m}(t) \left(\chi_{n,m,m'}^{ee}(t)\right)^2, \tag{11}$$

where $e_{n,m}^{e}$ is the energy consumption coefficient of edge server.

2.2.2 Cloud Data Processing

If edge-cloud migration is implemented, the cloud data processing delay is given by

$$\tau_{n,m}^{proc}(t) = x_{n,m,m'}(t)\, \frac{\alpha_{n,m}(t)\, \gamma_{n,m}(t)}{\chi_{n,m,m'}^{ec}(t)}, m' = M+1, \tag{12}$$

where $\chi_{n,m,m'}^{ec}(t)$ is the computing resources allocated by the cloud server for $d_{m,n}$.

The cloud data processing energy consumption is calculated as

$$E_{n,m}^{proc}(t) = x_{n,m,m'}(t)\, e_{n,m}^{c} a_{n,m}(t)\, \gamma_{n,m}(t) \left(\chi_{n,m,m'}^{ec}(t)\right)^2, m' = M+1, \tag{13}$$

where $e_{n,m}^{c}$ is the energy consumption coefficient of cloud server.

2.3 Total Energy Consumption and Delay Model

The total delay of the cloud-edge collaborative processing is calculated by summing transmission delay, edge-edge migration delay, edge processing delay, edge-cloud migration delay, and cloud processing delay, i.e.,

$$\tau_{n,m}^{tot}(t) = \tau_{n,m}^{up}(t) + \tau_{n,m}^{ee}(t) + \tau_{n,m}^{pro}(t) + \tau_{n,m}^{ec}(t) + \tau_{n,m}^{proc}(t). \tag{14}$$

The total energy consumption is calculated by summing transmission energy consumption, edge-edge migration energy consumption, edge processing energy consumption, edge-cloud migration energy consumption, and cloud processing energy consumption, i.e.,

$$E_{n,m}^{tot}(t) = E_{n,m}^{up}(t) + E_{n,m}^{ee}(t) + E_{n,m}^{pro}(t) + E_{n,m}^{ec}(t) + E_{n,m}^{proc}(t). \tag{15}$$

3 Problem Formulation and Decomposition

To promote low-latency and energy-efficient data processing for multi-flow integrated energy aggregation dispatch, the objective is minimizing the long-term average weighted sum of total energy consumption and delay of data processing of all the devices over T slots. The edge-side service data migration selection, computing resource allocation, and device-side transmission power control are jointly optimized under long-term energy consumption constraint. Define the set of optimization variables as $\mathbf{Y}(t) = \left\{x_{n,m,m'}(t), P_{n,m}^{tran}(t), \chi_{n,m,m'}^{ee}(t)\right\}$. The problem is formulated as

$$\begin{aligned}
&\mathbf{P1:} \min_{\mathbf{Y}(t)} \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\tau_{n,m}^{tot}(t) + \alpha E_{n,m}^{tot}(t)\right)\\
&\text{s.t.} C_1\colon \sum_{m'=1}^{M+1} x_{n,m,m'}(t) = 1, \forall d_{m,n}(t) \in \mathcal{D}_m, \forall t \in \mathcal{T},\\
&C_2\colon P_{n,m}^{tran}(t) \in \mathcal{P}^{tran}, \forall d_{m,n} \in \mathcal{D}_m, \forall t \in \mathcal{T},\\
&C_3\colon \sum_{n=1}^{N}\sum_{m=1}^{M} \chi_{n,m,m'}^{ee}(t) \le \chi_{m',\max}(t), \forall s_{m'} \in \mathcal{S}, \forall t \in \mathcal{T},\\
&C_4\colon \sum_{t=1}^{T} E_{n,m}^{tot}(t) \le E_{n,m,\max}, \forall d_{m,n} \in \mathcal{D}_m.
\end{aligned} \tag{16}$$

α is the weight of total energy consumption. C_1 is the constraint of service data migration selection, which indicates that data packet can only be processed by one edge server or the cloud server. C_2 defines the transmission power constraint. C_3 is the available computing resource constraint of edge server, which denotes that the total resources allocated by edge server should not exceed the maximum available amount of its computing resources $\chi_{m',\max}(t)$. C_4 is the long-term energy consumption constraint, indicating that the total energy consumption over T slots should not exceed the threshold $E_{n,m,\max}$ to promote energy-efficient data processing.

It is hard to settle **P1** in polynomial time in view of the coupling between long-term energy consumption constraint with slot-based optimization. Specifically, due to the lack of future foresight information, the policy aimed to minimize the weighted sum per slot may not necessarily guarantee long-term constraint. For example, utilizing too much energy to reduce delay in the current slot results in less energy budget in the future, causing a larger possibility of long-term constraint violation. To develop a tractable solution, we decompose the coupling among slots based on virtual queue theory [24]. Specifically, virtual queuing theory is an effective tool for decoupling the original problem into independently solved subproblems. It turns the long-term energy constraint problem into the queue stability problem. In essence, the input to the virtual queue is the available energy and its output is

the consumed energy. The queue backlog denotes the energy deficit or surplus for each time slot. The time-averaged values of the queue inputs and queue outputs converge to two finite constants, i.e., a^{av} and b^{av}. The queue backlog is rate-stable if and only if $a^{av} \leq b^{av}$. If $a^{av} > b^{av}$, the queue backlog is equal to a^{av} minus b^{av}. Therefore, the queue backlog does not tend to infinity, which can guarantee the long-term constraints. Define the energy deficit virtual queue corresponding to IoT device $d_{m,n}$ as

$$H_{n,m}(t+1) = \max\left\{H_{n,m}(t) + E_{n,m}^{tot}(t) - \frac{E_{n,m,\max}}{T}\right\}. \tag{17}$$

Adopting Lyapunov optimization [25], **P1** is decomposed into a bunch of optimization problems per slot, i.e.,

$$\begin{aligned}
&\mathbf{P2:} \min_{\mathbf{Y}(t)} \sum_{m=1}^{M}\sum_{n=1}^{N}\left[V\left(\tau_{n,m}^{tot}(t) + \alpha E_{n,m}^{tot}(t)\right) + H_{n,m}(t)\, E_{n,m}^{tot}(t)\right] \\
&\text{s.t.} C_1: \sum_{m'=1}^{M+1} x_{n,m,m'}(t) = 1, \forall d_{m,n}(t) \in \mathcal{D}_m, \forall t \in \mathcal{T}, \\
&C_2: P_{n,m}^{tran}(t) \in \mathcal{P}^{tran}, \forall d_{m,n} \in \mathcal{D}_m, \forall t \in \mathcal{T}, \\
&C_3: \sum_{n=1}^{N}\sum_{m=1}^{M} \chi_{n,m,m'}^{ee}(t) \leq \chi_{m',\max}(t), \forall s_{m'} \in \mathcal{S}, \forall t \in \mathcal{T}, \\
&C_4': H_{n,m}(t) \text{ is mean rate stable}, \forall d_{m,n} \in \mathcal{D}_m,
\end{aligned} \tag{18}$$

where V is the weight coefficient to characterize the trade-off between the optimization objective and the stability of the queue. C_4' indicates the queue stability constraint of $H_{n,m}(t)$.

4 Cloud-Edge Collaborative FGAN-Based Communication and Computing Resource Scheduling Algorithm for Multi-Flow Integrated Energy Aggregation Dispatch

To address the decomposed problem **P2**, we construct Markov decision processes (MDPs), and propose the cloud-edge collaborative FGAN-based communication and computing resource scheduling algorithm for multi-flow integrated energy aggregation dispatch. The proposed algorithm integrates federated learning with GAN to leverage global insights to improve the convergence and optimization performance of resource scheduling in multi-flow integrated energy aggregation dispatch data processing. In addition, the proposed algorithm utilizes the number of constraint violation occurrences to dynamically adjust the learning rate and network parameter weights to achieve constraint violation sensitiveness and improve accuracy and convergence speed. Compared to deep convolutional GAN (DCGAN), wasserstein GAN (WGAN) and progressive growing of GAN (PGGAN), FGAN reduces the risk of data leakage and reduces communication overhead by performing local model training and transmitting only local model parameters to the cloud server.

4.1 MDP

Firstly, we model the formulated problem as MDPs in the following:

State: In slot t, the state space of edge server s_m includes the size and computation complexity of device packet data, as well as energy deficit virtual queue backlog, i.e.,

$$\mathbf{O}_m(t) = \left\{a_{n,m}(t), H_{n,m}(t), \gamma_{n,m}(t), |d_{m,n} \in \mathcal{D}_m\right\}. \tag{19}$$

Action: In slot t, the action space incorporates optimization variables of service data migration, transmission power control, and computing resource allocation. Specially, the edge server computing

resource is divided into Q levels, i.e., $\chi^{ee}_{n,m,m'}(t) \in \left\{\chi^{ee}_{m',1}, \ldots, \chi^{ee}_{m',q'}, \chi^{ee}_{m',Q}\right\}$, where $\chi^{ee}_{m',1}$ and $\chi^{ee}_{m',Q}$ represent the minimum and maximum computing resources. $\chi^{ee}_{m',q} = \dfrac{(q-1)\left(\chi^{ee}_{m',Q} - \chi^{ee}_{m',1}\right)}{Q-1}$ is the q-th level of computing resources. The action space is given by

$$\mathbf{A}_m(t) = \left\{\mathbf{x}_m(t), \mathbf{p}^{tran}_m(t), \mathbf{X}^{ee}_m(t)\right\}, \tag{20}$$

where $\mathbf{x}_m(t) = \left\{x_{n,m,m'}(t) \,|d_{m,n}(t) \in \mathcal{D}_m(t)\right\}$, $\mathbf{p}^{tran}_m(t) = \left\{P^{tran}_{n,m}(t) \,|d_{m,n} \in \mathcal{D}_m\right\}$ and $\mathbf{X}^{ee}_m(t) = \left\{\chi^{ee}_{n,m,m'}(t) \,|d_{m,n} \in \mathcal{D}_m\right\}$.

Reward function: The reward function is designed as the negative optimization objective of problem **P2**, i.e.,

$$\Pi_m(t) = -\sum_{n=1}^{N} V\left(\tau^{tot}_{n,m}(t) + \alpha E^{tot}_{n,m}(t)\right) + \sum_{n=1}^{N} H_{n,m}(t) E^{tot}_{n,m}(t). \tag{21}$$

4.2 Coud-Edge Collaborative FGAN with Long-Term Constraint Violation Sensitiveness

Traditional GAN algorithms show slow convergence and inferior optimality without global knowledge. To address this issue, we augment traditional GAN with the federated learning framework and develop a novel cloud-edge collaborative FGAN algorithm with long-term constraint violation sensitiveness, the framework of which is shown in Fig. 2. The proposed algorithm provides edge servers with insights of entire network based on cloud-empowered global aggregation. This significantly improves the global optimality searching capability of traditional GAN and avoids falling into local optimal dilemmas. Each edge server maintains an actor-critic based GAN generator and a GAN discriminator. In the GAN generator, the actor network is used to observe states and learn scheduling policies with policy gradients, and the critic network learns the state value function to assist in policy update. The training purpose of GAN generator is to improve its capability of generating the resource scheduling strategy and to confuse GAN discriminator in a best-effort way. The GAN discriminator outputs the evaluation results and guides the update of the GAN generator according to the state-action pairs of the GAN generator. The training purpose of GAN discriminator is to improve its ability to distinguish the expert policy from the resource scheduling strategy generated by GAN generator. Thus, the edge server continuously optimizes the relationship between GAN generator and GAN discriminator through mutual gaming. At the end of each slot, each server transmits the GAN model parameters to the cloud server, which aggregates the parameters and distributes them to edge servers to guide the learning of the optimal scheduling decision under global information. The proposed algorithm improves the accuracy of the traditional reinforcement learning actor-critic structure for policy generation by employing a GAN network. Compared to deep Q-network (DQN), deep actor-critic (DAC) and deep deterministic policy gradient (DDPG), the proposed algorithm combines federated learning, which not only provides the necessary global information for GAN to accelerate its convergence, but also enables model training on various distributed endpoints, thus greatly reducing the cost of data transmission in data centers. Meanwhile, the proposed algorithm solves the problem that DQN, DAC and DDPG are not applicable to distributed scenarios, such as low-voltage distribution networks, and proves its effectiveness in dealing with distributed training scenarios.

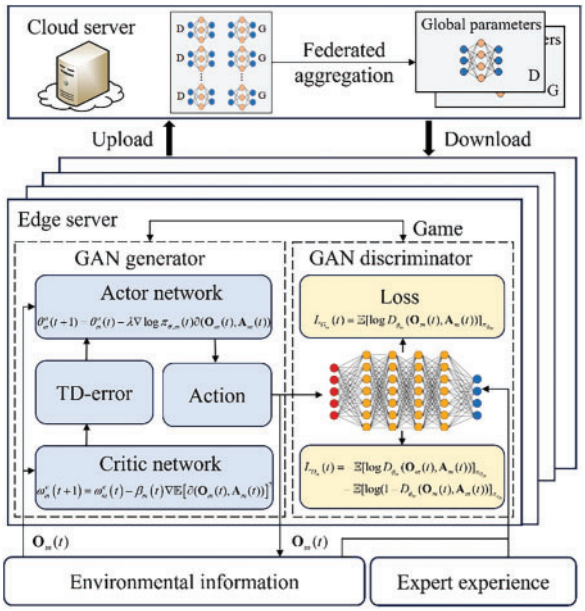


Figure 2: Framework of cloud-edge collaborative FGAN with long-term constraint violation sensitiveness

4.2.1 Actor-Critic Based GAN Generator

For the GAN generator of each edge server, the input and output of the actor network are the state space $\mathbf{O}_m(t)$ and the strategy $\pi_{\theta,m}(t)$, i.e., the probability distribution of action under input $\mathbf{O}_m(t)$. The edge server executes the action with the maximum probability and obtain reward $\Pi_m(t)$. Then, the edge server transfers to the next state space $\mathbf{O}_m(t+1)$. The output $\pi_{\theta,m}(t)$ is utilized to update actor network parameters through gradient descent as

$$\theta_m^a(t+1) = \theta_m^a(t) - \lambda \nabla \log \pi_{\theta,m}(t)\, \partial\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right). \tag{22}$$

$\theta_m^a(t)$ is the parameters of the actor network. $\nabla(\cdot)$ is the gradient function and λ is the learning rate.

The policies generated by the actor network are evaluated by the critic network, the input of which is the states $\mathbf{O}_m(t)$ and $\mathbf{O}_m(t+1)$. The output is the value functions $V_m(\mathbf{O}_m(t))$ and $V_m(\mathbf{O}_m(t+1))$, which measure the expected return of $\mathbf{O}_m(t)$ and $\mathbf{O}_m(t+1)$ when following the current strategy. Temporal difference error (TD-error) is adopted to evaluate the deviation between the expected return and actual reward, which is calculated as

$$\partial\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right) = \Pi_m(t) + \gamma V_m(\mathbf{O}_m(t+1)) - V_m(\mathbf{O}_m(t)), \tag{23}$$

where γ is the decay factor to adjust the degree of reference to future decisions. Then, update the critic network parameter $\omega_m^c(t)$ as

$$\omega_m^c(t+1) = \omega_m^c(t) - \beta_m(t)\, \nabla \mathbb{E}\left[\partial\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right]^2. \tag{24}$$

$\mathbb{E}[\cdot]$ represents the expectation. $\beta_m(t)$ represents the learning rate, which adjusts the updating amplitude of parameters. A larger learning rate indicates a larger updating amplitude of $\omega_m^c(t)$. To further improve parameter updating accuracy and convergence speed, we develop a dynamic adjustment method of learning rate, where $\beta_m(t)$ is adjusted in accordance with constraint violation occurrences.

Define $\mathbb{I}[\cdot]$ as the indicator function. When $E_{n,m}^{up}(t) > \frac{E_{n,m,\max}}{T}$, $\mathbb{I}\left[E_{n,m}^{up}(t) > \frac{E_{n,m,\max}}{T}\right] = 1$, it represents that the long-term energy consumption constraint is violated. Otherwise, $\mathbb{I}\left[E_{n,m}^{up}(t) > \frac{E_{n,m,\max}}{T}\right] = 0$. Therefore, $\beta_m(t)$ is dynamically adjusted as

$$\beta_m(t) = \varphi\left[\kappa + \frac{1}{2}(1-\kappa)\left(1 + e^{\sum_{n=1}^{N}\mathbb{I}\left[E_{n,m}^{up}(t) > \frac{E_{n,m,\max}}{T}\right]}\right)^{-1}\right], \tag{25}$$

where φ represents the initial learning rate, and κ is the learning rate adjustment factor. If none constraint violation occurs, the learning rate remains unchanged as the initial value, i.e., $\beta_m(t) = \varphi$. On the other hand, if there exist numerous constraint violation occurrences, the learning rate is enlarged to accelerate learning speed of the generator network, thereby realizing constraint violation sensitive decision-making optimization.

4.2.2 GAN Discriminator

GAN discriminator takes the set of states $\mathbf{O}_m(t)$ as input, and the expert strategy and the action $\mathbf{A}_m(t)$ are output by GAN generator. Define $D_{\phi m}(\mathbf{O}_m(t), \mathbf{A}_m(t))$ as the output of the GAN discriminator. It represents the probability that the GAN discriminator distinguishes the policy generated by the generator from the expert policy in the expert knowledge base. When the GAN discriminator cannot effectively discriminate between the GAN generator policy and the expert policy, the GAN network training is completed. Therefore, the objective function of GAN discriminator is defined as

$$\begin{aligned}\min_{G_m}\max_{D_{\phi m}} V\left(G_m, D_{\phi m}\right) = {} & \mathbb{E}\left[\log\left(D_{\phi m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right)\right]_{\pi_{G_m(t)}} \\ & + \mathbb{E}\left[\log\left(1 - D_{\phi m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right)\right]_{\pi_{e_m(t)}}.\end{aligned} \tag{26}$$

G_m represents the generator network, $D_{\phi m}$ represents the discriminator network, $\pi_{G_m}(t)$ is the generator's generation policy, $\pi_{e_m}(t)$ is the expert strategy generated by the expert knowledge base, and $\phi_m(t)$ denotes the discriminator network parameters.

The GAN discriminator updates the GAN generator parameters and its own network parameters based on the discrimination results. For the generator, its objective is to produce a policy that the discriminator recognizes as an expert strategy, i.e., $D_{\phi m}(\mathbf{O}_m(t), \mathbf{A}_m(t))_{\pi_{G_m(t)}}$ tends to be 0. Therefore, the loss function that measures the goodness of the GAN generator strategy is defined as

$$L_{G_m}(t) = \mathbb{E}\left[\log D_{\phi m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right]_{\pi_{G_m(t)}}. \tag{27}$$

For the GAN discriminator, its objective is to distinguish between expert strategies and generated strategies from the generator, i.e., $D_{\phi m}(\mathbf{O}_m(t), \mathbf{A}_m(t))_{\pi_{G_m(t)}}$ tends to be 1 and $D_{\phi m}(\mathbf{O}_m(t), \mathbf{A}_m(t))_{\pi_{e_m(t)}}$ approaches 0. Therefore, the loss function that measures the goodness of the GAN discriminator is defined as

$$L_{D_m}(t) = -\mathbb{E}\left[\log D_{\phi m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right]_{\pi_{G_m(t)}} - \mathbb{E}\left[\log\left(1 - D_{\phi m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right)\right]_{\pi_{e_m(t)}}. \tag{28}$$

Traditional GAN algorithm adopts Jenson's Shannon (JS) divergence as the discriminator loss function. When there is no overlap between the GAN generator policy distribution and the expert policy distribution, JS divergence will be constant, resulting in gradient disappearance and making

the training difficult in the parameter update process. Thus, this paper builds upon traditional GAN by incorporating the Wasserstein distance to evaluate the deviation between generated and expert strategies. The Wasserstein distance is defined as

$$W\left(\pi_{G_m}(t), \pi_{e_m}(t)\right) = \sup_{\|f_D\|_L \leq 1} \mathbb{E}\left[\log D_{\phi_m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right]_{\pi_{G_m}(t)} - \mathbb{E}\left[\log D_{\phi_m}\left(\mathbf{O}_m(t), \mathbf{A}_m(t)\right)\right]_{\pi_{e_m}(t)}, \tag{29}$$

where $\| f_D \|_L \leq 1$ denotes that the function $f_D = \log(D_{\phi_m}(\mathbf{O}_m(t), \mathbf{A}_m(t))$ needs to follow 1-Lipschitz continuity, and the upper bound of the absolute value of its derivative is 1. $\sup(\cdot)$ denotes the solution of the supremum, i.e., the minimum upper bound.

4.2.3 Federated Learning Enabled GAN Parameter Updating

Federated learning is a distributed learning method that trains global parameters by sharing trained parameters among various edge servers instead of original dataset, which is introduced into GAN in this paper. Define $\omega^c(t)$, $\theta^a(t)$, and $\phi(t)$ as the global GAN generator actor network parameters, the global GAN generator critic network parameters, and the global GAN discriminator network parameters in slot t, respectively. Similarly, define $\omega_m^c(t)$, $\theta_m^a(t)$, and $\phi_m(t)$ as the GAN generator actor network parameters of s_m, the GAN generator critic network parameters of s_m, and the GAN discriminator network parameters of s_m in slot t, respectively. The cloud server distributes parameters $\omega^c(t)$, $\theta^a(t)$, and $\phi(t)$ to various edge servers. Each edge server uses these parameters to train its GAN parameters. The detailed procedures are shown in Algorithm 1, and are detailedly described in the following:

Algorithm 1: Coud-edge Collaborative FGAN with Long-Term Constraint Violation Sensitiveness

Initialization:

1: Initialize $a_{n,m}(t) = 0$, $H_{n,m}(t) = 0$, $\omega_m^c(t) = \omega^c(t)$, $\theta_m^a(t) = \theta^a(t)$ and $\phi_m(t) = \phi(t)$.
2: **while** $W\left(\pi_{G_m}(t), \pi_{e_m}(t)\right) > W$ **do**
3: Each $s_m \in \mathcal{S}$ downloads global parameters $\omega^c(t)$, $\theta^a(t)$, and $\phi(t)$.
4: Each $s_m \in \mathcal{S}$ takes state $\mathbf{O}_m(t)$ as input of GAN generator actor network, and outputs the policy $\pi_{\theta,m}(t)$.
5: Each $s_m \in \mathcal{S}$ executes the action $\mathbf{A}_m(t)$ based on policy $\pi_{\theta,m}(t)$ and obtains reward $\Pi_m(t)$.
6: Update $\mathbf{O}_m(t)$ to $\mathbf{O}_m(t+1)$.
7: Update $\theta_m^a(t)$ based on (22).
8: Each $s_m \in \mathcal{S}$ takes $\mathbf{O}_m(t)$ to $\mathbf{O}_m(t+1)$ as input of GAN generator critic network, and outputs the value functions $V_m(\mathbf{O}_m(t))$ and $V_m(\mathbf{O}_m(t+1))$
9: Calculate $\partial(\mathbf{O}_m(t), \mathbf{A}_\mathbf{m}(t))$ based on (23)
10: Update $\beta_m(t)$ based on (25).
11: Update $\omega_m^c(t)$ based on (25).
12: Each $s_m \in \mathcal{S}$ takes the set of states $\mathbf{O}_m(t)$, the expert strategy, and the action $\mathbf{A}_\mathbf{m}(t)$ as input of GAN discriminator, and outputs $D(\mathbf{O}_m(t), \mathbf{A}_m(t))$.
13: Update $L_{G_m}(t)$, $L_{D_m}(t)$, and $W\left(\pi_{G_m}(t), \pi_{e_m}(t)\right)$ based on (27)–(29).
14: Each $s_m \in \mathcal{S}$ uploads GAN parameters $\omega_m^c(t+1)$, $\theta_m^a(t+1)$, and $\phi_m(t+1)$ to cloud server.
15: Perform cloud server aggregation as (30).
16: $t = t + 1$.

Step 1: Each edge server uses the distributed global parameters $\omega^c(t)$, $\theta^a(t)$, and $\phi(t)$ to update its GAN parameters based on (26) and (29).

Step 2: After edge-side updating, each edge server uploads GAN parameters $\omega_m^c(t+1)$, $\theta_m^a(t+1)$, and $\phi_m(t+1)$ to the cloud server. Then, the cloud server performs aggregation to obtain the global GAN parameters by collecting parameters from all edge servers, which is given by

$$\begin{cases} \omega^c(t+1) = \frac{1}{\sum_{m=1}^{M}\rho_m(t)} \sum_{m=1}^{M} \rho_m(t)\,\omega_m^c(t+1), \\ \theta^a(t+1) = \frac{1}{\sum_{m=1}^{M}\rho_m(t)} \sum_{m=1}^{M} \rho_m(t)\,\theta_m^a(t+1), \\ \phi(t+1) = \frac{1}{\sum_{m=1}^{M}\rho_m(t)} \sum_{m=1}^{M} \rho_m(t)\,\phi_m(t+1), \end{cases} \tag{30}$$

where $\rho_m(t)$ represents the weight coefficients assigned to network parameters of different edge servers. A larger weight $\rho_m(t)$ indicates that the parameters of s_m contribute more to global parameter aggregation. To further improve the accuracy of global parameter aggregation, we design a dynamic adjustment mechanism of weight related to the constraint violation occurrences. Similar to (25), the indicator function $\mathbb{I}\left[E_{n,m}^{tot}(t) > \frac{E_{n,m,\max}}{T}\right]$ is utilized to adjust $\rho_m(t)$, which is given by

$$\rho_m(t) = \frac{num_{\max}^E(t) - \sum_{n=1}^{N}\mathbb{I}\left[E_{n,m}^{up}(t) > \frac{E_{n,m,\max}}{T}\right]}{num_{\max}^E(t) - num_{\min}^E(t)} \tag{31}$$

where $num_{\max}^E(t)$ and $num_{\min}^E(t)$ represent the maximum and minimum numbers of energy consumption constraint violations per time slot. If devices managed by s_m occur less constraint violation, it means that its network parameters are updated well. Its weight in global parameter aggregation needs to be increased to improve the aggregation performance. On the contrary, if devices managed by s_m exist numerous constraint violation occurrences, its weight needs to be decreased.

Step 3: Repeat steps 1 and 2 until the GAN generator policy for each edge server is close to the same as the expert policy, i.e., $W\left(\pi_{G_m}(t), \pi_{e_m}(t)\right) \leq W_{\min}$, where $W_{\min}$ is the preset training threshold.

4.3 Computation Complexity

The proposed algorithm is divided into GAN and federated learning enabled GAN parameters updating. The computation complexity of the GAN generator is $O(MN)$ and the computation complexity of the GAN discriminator is $O(MN)$. Therefore, the computation complexity of GAN is $O(2MN)$. The computation complexity of the federated learning local model update is $O(K \cdot MN)$, where K is the number of local updates. The computation complexity of federated learning global model aggregation is $O(M)$. Therefore, the computation complexity of federated learning is $O(K \cdot MN) + O(M)$. The computation complexity of the proposed algorithm is $2O(MN) + O(K \cdot MN) + O(M)$.

5 Simulation Results

The effectiveness of the cloud-edge collaborative FGAN algorithm with long-term constraint violation sensitiveness is verified through simulations. We consider a multi-flow integrated energy aggregation dispatch scenario based on IEEE 33 bus [26], which is shown in Fig. 3. The scenario consists of 25 devices, 5 edge servers and 1 cloud server. Other relevant parameters are specified in Table 1 [22,27]. Three comparison algorithms are utilized. The federated deep actor-critic (DAC)

based cloud-edge collaborative resource scheduling algorithm (FDAC) [28], the GAN based cloud-edge collaborative resource scheduling algorithm (GAN) [29], and the DDPG based cloud-edge collaborative resource scheduling algorithm (DDPGRS) [30] are adopted as comparison algorithms. In FDAC, the edge server uses the traditional DAC algorithm to generate communication and computing resource scheduling decisions, and the cloud server performs model aggregation and parameter dissemination. In GAN, each edge server establishes an independent GAN to learn communication and computing resource scheduling decisions, and the cloud only performs data processing rather than global aggregation. Both FDAC and GAN do not take into account the long-term constraint of energy consumption. In DDPGRS, each edge server connected to the central cloud server via a fiber connection utilizes the actor part of DDPG to search the optimal data offloading strategy and energy consumption control of the device. However, DDPGRS does not take into account the edge-edge migration.

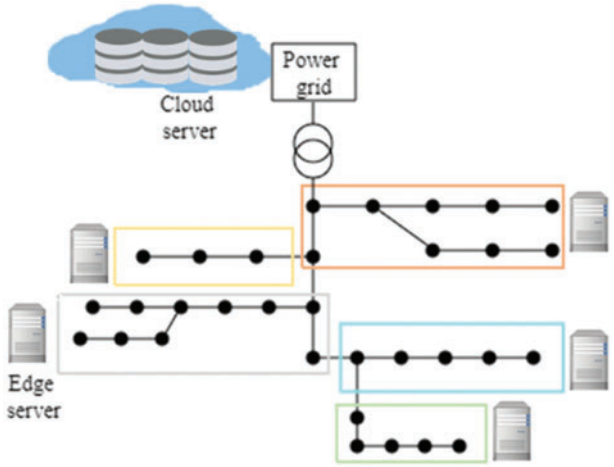


Figure 3: IEEE 33 bus model based on simulation scenario

Table 1: Simulation parameters

Parameter	Value
T	200
M	6
δ^2	−114 dBm
N	25
K	5
λ	0.99
α	0.2
$B^{up}_{n,m}(t)$	1 MHz
$P^{tran}_{n,m,\min}$	0.1 W

(Continued)

Table 1 (continued)

Parameter	Value
$P^{tran}_{n,m,\max}$	0.5 W
$\tau^{ec}_{m,m'}$	[0.15 0.25] s
$\tau^{ee}_{m,m'}$	[0.1 0.14] s
$\gamma_{n,m}$	$[7 \times 10^6, 1.1 \times 10^7]$ cycles/Mbit
$a_{n,m}(t)$	[0.8, 1.2] Mbits
$\chi_{m',\max}$	1.2×10^9 CPU cycles/s

Fig. 4 demonstrates the average weighted sum of total energy consumption and delay *vs.* time slot. The proposed algorithm has the minimum weighted sum value and smaller fluctuations. When $t = 200$, the proposed algorithm reduces the weighted sum by 33.24%, 46.74% and 23.47% compared to FDAC, GAN and DDPGRS, respectively. The reason is that the proposed algorithm utilizes federated learning to aggregate model parameters across edge servers, a strategy whose core advantage lies in its ability to provide a global convergence point of information to guide the GANs to converge more efficiently. In addition, the cloud and edge servers cooperate to perform network training and parameter interaction to jointly optimize communication resource scheduling, such as transmission power, as well as computing resource scheduling, including data migration and computing resource allocation.

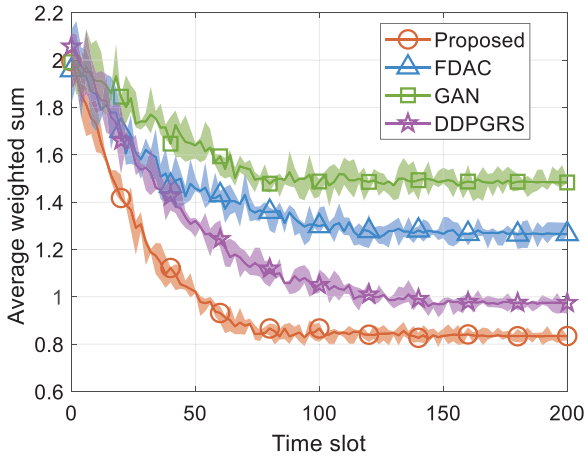


Figure 4: Average weighted sum of total energy consumption and delay *vs.* time slot

Fig. 5 demonstrates the average weighted sum of total energy consumption and delay for 50 simulations. Compared to the FDAC, GAN and DDPGRS, the proposed algorithm reduces the median of the average weighted sum of total energy consumption and delay by 35.71%, 42.65% and 26.95%, respectively. Meanwhile, the proposed algorithm has the smallest mean and variance of the average weighted sum of total energy consumption and delay, indicating the reliability and generalizability of the findings.

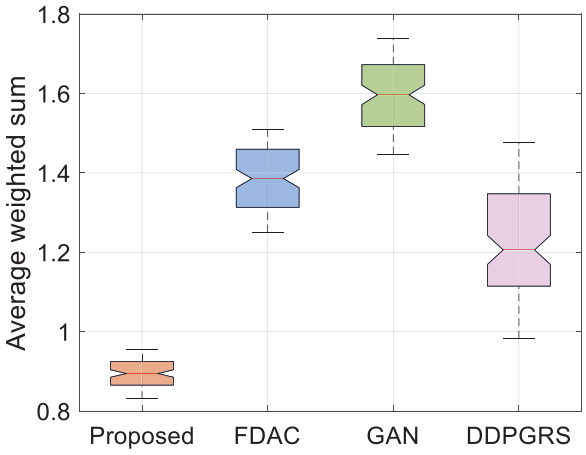
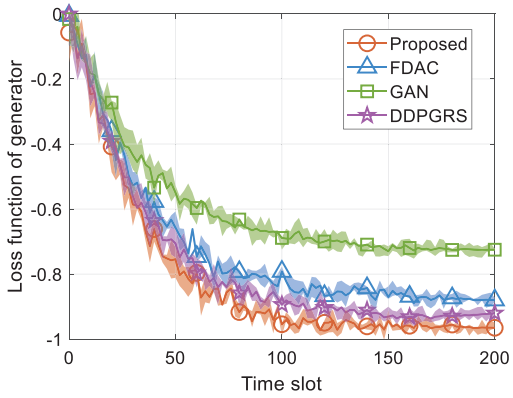


Figure 5: Average weighted sum of total energy consumption and delay for 50 simulations

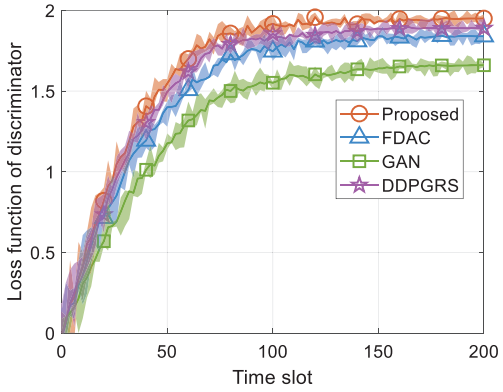
Fig. 6 shows the loss functions *vs.* time slot, respectively. The proposed algorithm has the fastest convergence, the lowest loss function of generator, and the highest loss function of discriminator. Compared to FDAC, GAN and DDPGRS, the loss function of generator achieved by the proposed algorithm is decreased by 9.66%, 33.10%, and 12.37%. The rationale behind this is that the proposed algorithm adopts the Wasserstein distance rather than the traditional JS divergence to calculate loss function. During model training, the Wasserstein distance significantly improves the accuracy of the model in capturing the details of the data distribution, which in turn effectively contributes to the generalization performance of the model. When the Wasserstein distance is adopted as a loss function, it motivates the generative model to produce outputs that are closer to the real data distribution.

Fig. 7 shows the energy consumption virtual queue backlogs of different algorithms. The red line denotes the median. Compared to the FDAC, GAN and DDPGRS, the proposed algorithm reduces the median of the energy consumption virtual deficit queue backlog by 49.90%, 62.55% and 34.64%, respectively. Moreover, the proposed algorithm decreases the virtual deficit queue backlog deviation by 75.69%, 68.23% and 35.86%. The reason is that the proposed algorithm considers energy consumption virtual deficit queues and applies them in the optimization objective, so as to optimize the scheduling policy according to the urgency of tasks and real-time availability of resources, which effectively reduces the service delay as well as the system energy consumption. In addition, the algorithm considers the sensitivity to constraint violations when dynamically adjusting the learning rate and global aggregation weights to minimize the occurrence of constraint violations.

Fig. 8 demonstrates the trade-off between the total energy consumption and the total delay under different weight values of α. The total energy consumption denotes the sum of transmission energy consumption, edge-edge migration energy consumption, edge processing energy consumption, edge-cloud migration energy consumption, and cloud processing energy consumption for all devices across a span of 200 time slots. From the figure, it is shown that as the weight of energy consumption increases from 0.4 to 2, the total energy consumption is declined by 38.16%, and the total delay is increased by 19.82%. It indicates that as the weight factor increases, the proposed algorithm gradually pays more attention to energy consumption reduction at the cost of increased delay. Therefore, the value of α should be carefully determined in accordance with energy dispatch service requirements to enable a well balance between energy consumption and delay.



(a) Loss function of generator *vs.* time slot



(b) Loss function of discriminator *vs.* time slot

Figure 6: Loss functions *vs.* time slot

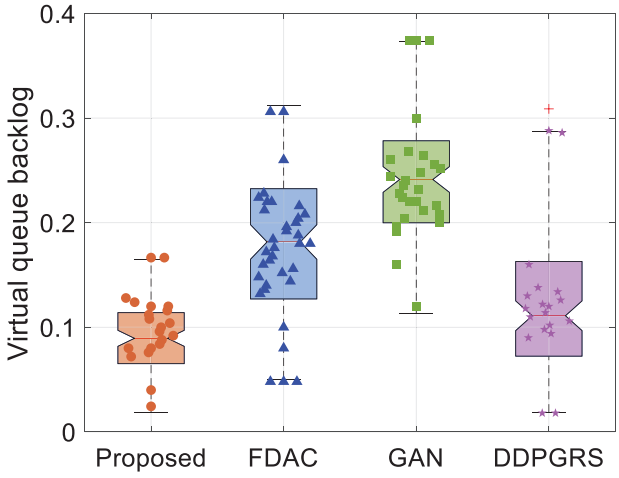


Figure 7: Energy consumption virtual queue backlogs of different algorithms

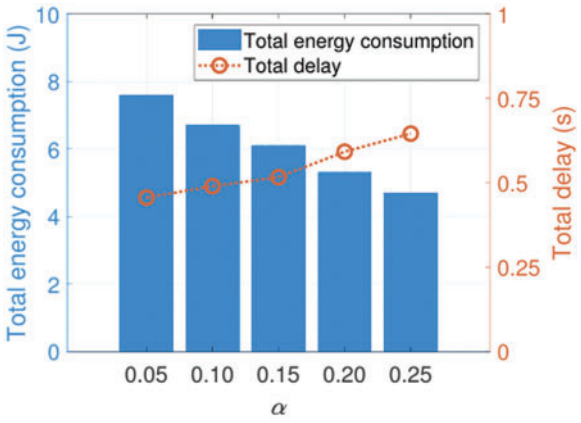


Figure 8: Total energy consumption and total delay *vs.* the weight α

Fig. 9 shows delay composition and migration composition *vs.* edge server computing resources. As the edge server computing resources increase from 3×10^{10} to 7×10^{10} CPU cycles/s, the proportion of edge-side processing delay is reduced from 8.92% to 5.08%. Meanwhile, the proportions of both the edge-edge migration and edge-cloud migration are reduced from 53.57% to 37.79%. This is because the increase of edge server computing resources facilitates lower edge-side processing delay, and the data from devices are more inclined to be processed locally by edge server rather than relying on edge-edge migration and edge-cloud migration.

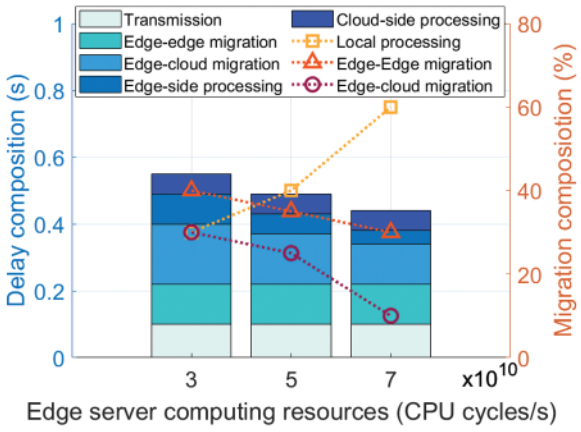


Figure 9: Delay composition and migration composition *vs.* edge server computing resources

Fig. 10 shows the delay composition and migration composition *vs.* data computation complexity. As the data computation complexity increases from 7×10^6 to 11×10^6 cycles/Mbit, the local edge server becomes heavy loaded, and the proportion of local processing delay is decreased by 57.22%. This enforces that more data are migrated to other light-loaded edge servers or the cloud server whose computing resources are sufficient. Therefore, edge-edge migration proportion and edge-cloud migration proportion are increased by 39.26% and 71.41%. From the figure, it is obvious that the proposed algorithm demonstrates superior performance in improving workload balance by dynamically adjusting the tradeoff among local processing, edge-edge migration and edge-cloud

migration. In addition, the incorporation of federated learning provides global insight of the entire network for each edge server to optimize its decision making and local parameter updating of GAN.

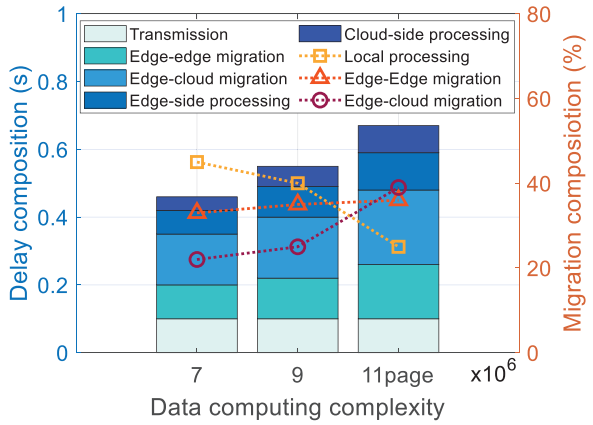


Figure 10: Delay composition and migration composition *vs.* data computation complexity

Table 2 shows the computation complexity of different algorithms. Although federated learning introduces additional communication overhead, the computation complexity of the proposed algorithm is still in the same order of magnitude as FDAC, GAN. The computation complexity of the proposed algorithm is slightly higher than FDAC, GAN, and DDPGRS, but the weighted sum of the proposed algorithms is reduced by 33.24%, 46.74%, and 23.47% compared to FDAC, GAN, and DDPGRS, respectively.

Table 2: Comparison of the computation complexity

Algorithm	Computation complexity (cycles/Mbit)
Proposed	9×10^6
FDAC	6×10^6
GAN	7×10^6
DDPG	13×10^6

6 Conclusion

This paper addresses the joint optimization challenge of communication and computing resource scheduling for multi-flow integrated energy aggregation dispatch. The study introduces a cloud-edge collaborative FGAN algorithm that is sensitive to long-term constraint violations to facilitate energy-efficient data processing with reduced latency, even under stringent energy consumption constraints. This method significantly enhances energy efficiency and response times, offering a robust solution for the real-world implementation of multi-flow integrated energy aggregation dispatch systems. It also presents innovative strategies for adapting to diverse computational resources and data complexities. When benchmarked against FDAC, GAN and DDPGRS, the proposed FGAN algorithm achieves a 33.24%, 46.74% and 34.64% reduction in the average weighted sum of energy consumption and latency, respectively, and lowers the average energy consumption virtual queue deficit backlogs by 49.90% and 62.55% and 35.86%. Simulation results further reveal its capability to dynamically adjust migrations

between edge devices and from edge to cloud, in response to fluctuations in available computing resources and data processing complexities, thereby ensuring workload balance.

However, the proposed algorithm still has some potential limitations. For instance, it does not account for certain unexpected situations and possible attacks, posing risks of privacy breaches, which may result in low scheduling accuracy. Smart contracts and blockchain technology enable data and transactions to occur on decentralized networks, reducing the risk of single points of failure and enhancing the reliability and security of the system. Future research efforts could combine smart contracts with blockchain to improve security.

Acknowledgement: The author would like to thank the Electric Power Dispatching Control Center of Guangdong Power Grid Co., Ltd. for supporting this work.

Funding Statement: This work was supported by China Southern Power Grid Technology Project under Grant 03600KK52220019 (GDKJXM20220253).

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The author declares that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Wu, V. K. N. Lau, D. H. K. Tsang, L. P. Qian, and L. Meng, "Optimal energy scheduling for residential smart grid with centralized renewable energy source," *IEEE Syst. J.*, vol. 8, no. 2, pp. 562–576, Jun. 2014. doi: 10.1109/JSYST.2013.2261001.

[2] Y. Liu, S. Xie, Q. Yang, and Y. Zhang, "Joint computation offloading and demand response management in mobile edge network with renewable energy sources," *IEEE Trans. Vehicular Technol.*, vol. 69, no. 12, pp. 15720–15730, Dec. 2020. doi: 10.1109/TVT.2020.3033160.

[3] L. Lei, Y. Tan, G. Dahlenburg, W. Xiang, and K. Zheng, "Dynamic energy dispatch based on deep reinforcement learning in IoT-driven smart isolated microgrids," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7938–7953, May 2021. doi: 10.1109/JIOT.2020.3042007.

[4] J. Zhang, Q. Yan, X. Zhu, and K. Yu, "Smart industrial IoT empowered crowd sensing for safety monitoring in coal mine," *Digit. Commun. Netw.*, vol. 9, no. 2, pp. 296–305, Apr. 2023. doi: 10.1016/j.dcan.2022.08.002.

[5] P. Srikantha and D. Kundur, "Intelligent signal processing and coordination for the adaptive smart grid: An overview of data-driven grid management," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 82–102, May 2019. doi: 10.1109/MSP.2018.2877001.

[6] D. Gan, X. Ge, and Q. Li, "An optimal transport-based federated reinforcement learning approach for resource allocation in cloud-edge collaborative IoT," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2407–2419, Jan. 15, 2024. doi: 10.1109/JIOT.2023.3292368.

[7] Z. Zhou *et al.*, "Blockchain-based secure and efficient secret image sharing with outsourcing computation in wireless networks," *IEEE Trans. Wirel. Commun.*, vol. 23, no. 1, pp. 423–435, 2024. doi: 10.1109/TWC.2023.3278108.

[8] H. Liao *et al.*, "Cloud-edge-device collaborative reliable and communication-efficient digital twin for low-carbon electrical equipment management," *IEEE Trans. Ind. Inform.*, vol. 19, no. 2, pp. 1715–1724, Feb. 2023. doi: 10.1109/TII.2022.3194840.

[9] J. H. Syu, J. C. W. Lin, G. Srivastava, and K. Yu, "A comprehensive survey on artificial intelligence empowered edge computing on consumer electronics," *IEEE Trans. Consum. Electron.*, vol. 69, no. 4, pp. 1023–1034, 2023. doi: 10.1109/TCE.2023.3318150.

[10] F. Fang and X. Wu, "A win-win mode: The complementary and coexistence of 5G networks and edge computing," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 3983–4003, Mar. 15, 2021. doi: 10.1109/JIOT.2020.3009821.

[11] Z. Yang *et al.*, "Differentially private federated tensor completion for cloud-edge collaborative AIoT data prediction," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 256–267, Jan. 1, 2024. doi: 10.1109/JIOT.2023.3314460.

[12] F. Zeng, K. Zhang, L. Wu, and J. Wu, "Efficient caching in vehicular edge computing based on edge-cloud collaboration," *IEEE Trans. Vehicular Technol.*, vol. 72, no. 2, pp. 2468–2481, Feb. 2023. doi: 10.1109/TVT.2022.3213130.

[13] A. T. Z. Kasgari, W. Saad, M. Mozaffari, and H. V. Poor, "Experienced deep reinforcement learning with generative adversarial networks (GANs) for model-free ultra reliable low latency communication," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 884–899, Feb. 2021. doi: 10.1109/TCOMM.2020.3031930.

[14] R. K. Gupta, S. Mahajan, and R. Misra, "Resource orchestration in network slicing using GAN-based distributional deep Q-network for industrial applications," *J. Supercomput.*, vol. 79, no. 5, pp. 5109–5138, Oct. 2022. doi: 10.1007/s11227-022-04867-9.

[15] Y. Hua, R. Li, Z. Zhao, H. Zhang, and X. Chen, "GAN-based deep distributional reinforcement learning for resource management in network slicing," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9014217.

[16] F. Naeem, S. Seifollahi, Z. Zhou, and M. Tariq, "A generative adversarial network enabled deep distributional reinforcement learning for transmission scheduling in internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4550–4559, Jul. 2021. doi: 10.1109/TITS.2020.3033577.

[17] S. Zhang, Z. Yao, H. Liao, Z. Zhou, Y. Chen and Z. You, "Endogenous security-aware resource management for digital twin and 6G edge intelligence integrated smart park," *China Commun.*, vol. 20, no. 2, pp. 46–60, Feb. 2023. doi: 10.23919/JCC.2023.02.004.

[18] C. Xu, R. Xia, Y. Xiao, Y. Li, G. Shi and K. C. Chen, "Federated traffic synthesizing and classification using generative adversarial networks," in *ICC 2021-IEEE Int. Conf. Commun.*, Montreal, QC, Canada, 2021, pp. 1–6. doi: 10.1109/ICC42927.2021.9500866.

[19] E. Tomiyama, H. Esaki, and H. Ochiai, "WAFL-GAN: Wireless ad hoc federated learning for distributed generative adversarial networks," in *2023 15th Int. Conf. Knowl. Smart Technol. (KST)*, Phuket, Thailand, 2023, pp. 1–6. doi: 10.1109/KST57286.2023.10086811.

[20] H. Sui, X. Sun, J. Zhang, B. Chen, and W. Li, "Multi-level membership inference attacks in federated learning based on active GAN," *Neural Comput. Appl.*, vol. 35, no. 23, pp. 17013–17027, Apr. 2023. doi: 10.1007/s00521-023-08593-y.

[21] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma and X. Cui, "IFL-GAN: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10502–10515, Dec. 2023. doi: 10.1109/TNNLS.2022.3167482.

[22] G. Laguna-Sanchez and M. Lopez-Guerrero, "On the use of alpha-stable distributions in noise modeling for PLC," *IEEE Trans. Power Deliv.*, vol. 30, no. 4, pp. 1863–1870, Aug. 2015. doi: 10.1109/TPWRD.2015.2390134.

[23] Z. Zhou, Y. Guo, Y. He, X. Zhao, and W. M. Bazzi, "Access control and resource allocation for M2M communications in industrial automation," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3093–3103, May 2019. doi: 10.1109/TII.2019.2903100.

[24] M. Wasim and D. S. Naidu, "Lyapunov function construction using constrained least square optimization," in *IECON, 2022–48th Annual Conference of the IEEE Industrial Electronics Society*, Brussels, Belgium, 2022, pp. 1–5. doi: 10.1109/IECON49645.2022.9968442.

[25] J. Zhang, S. Guo, J. Guo, D. Zeng, J. Zhou and A. Y. Zomaya, "Towards data-independent knowledge transfer in model-heterogeneous federated learning," *IEEE Trans. Comput.*, vol. 72, no. 10, pp. 2888–2901, Oct. 2023. doi: 10.1109/TC.2023.3272801.

[26] J. Liu, P. Li, G. Wang, Y. Zha, J. Peng and G. Xu, "A multitasking electric power dispatch approach with multi-objective multifactorial optimization algorithm," *IEEE Access*, vol. 8, pp. 155902–155911, 2020. doi: 10.1109/ACCESS.2020.3018484.

[27] B. Kar, W. Yahya, Y. D. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud-edge–fog systems: A survey," *IEEE Commun. Surv. Tut.*, vol. 25, no. 2, pp. 1199–1226, Secondquarter 2023. doi: 10.1109/COMST.2023.3239579.

[28] Z. Su *et al.*, "Secure and efficient federated learning for smart grid with edge-cloud collaboration," *IEEE Trans. Ind. Inform.*, vol. 18, no. 2, pp. 1333–1344, Feb. 2022. doi: 10.1109/TII.2021.3095506.

[29] R. Yan, Y. Yuan, Z. Wang, G. Geng, and Q. Jiang, "Active distribution system synthesis via unbalanced graph generative adversarial network," *IEEE Trans. Power Syst.*, vol. 38, no. 5, pp. 4293–4307, Sep. 2023. doi: 10.1109/TPWRS.2022.3212029.

[30] H. Hu, D. Wu, F. Zhou, X. Zhu, R. Q. Hu and H. Zhu, "Intelligent resource allocation for edge-cloud collaborative networks: A hybrid DDPG-D3QN approach," *IEEE Trans. Vehicular Technol.*, vol. 72, no. 8, pp. 10696–10709, Aug. 2023. doi: 10.1109/TVT.2023.3253905.