A Differential Evolution Offloading Strategy for Latency and Privacy Sensitive Tasks with Federated Local-edge-cloud Collaboration

YISHAN CHEN, JiangXi University of Science and Technology, Ganzhou, China
WEI LI, JiangXi University of Science and Technology, Ganzhou, China
JUNHONG HUANG, JiangXi University of Science and Technology, Ganzhou, China
HONGHAO GAO*, Shanghai University, Shanghai, China
SHUIGUANG DENG*, Zhejiang University, Hangzhou, China

Due to an explosive growth in mobile devices and the rapid evolution of wireless communication technologies, local-edge-cloud computing is becoming an attractive solution for providing a higher-quality service by exploiting the multi-computation power of mobile devices, edge servers and cloud. However, as the tasks are latency and privacy sensitive, highly credible task offloading becomes a crucial problem in a local-edge-cloud orchestrated computing system. In this paper, we study the computation offloading problem for latency and privacy sensitive tasks in a hierarchical local-edge-cloud network by using federated learning method. Our goal is to minimize the operational time of latency-sensitive tasks requested by mobile devices that have data privacy concerns, while each task can be executed under local, edge or cloud computing mode with no need to rely on privacy data. We first build system models to analyze the latency incurred under different computing modes, and then develop a constrained optimization problem to minimize the latency consumed by the federated offloading collaboration. A Hierarchical Federated Averaging method based on Differential Evolution algorithm (HierFAVG-DE) is proposed for solving the problem in-hand, and extensive simulations are conducted to verify the superiority of our approach.

Additional Key Words and Phrases: Latency, Privacy, Offloading, Federated Learning, Differential Evolution

1 INTRODUCTION

Due to high explosion of devices and rapid evolution of wireless communication technologies, local-edge-cloud computing is becoming an attractive solution for providing a higher-quality service by exploiting the multi-computation power of mobile devices, edge servers and cloud, especially in the field of Smart City [1], Internet of Vehicles (IoV) [16] and Industrial Internet of Things (IIoT) [22, 37, 38]. Intelligent Transportation [28] may be a good example using local-edge-cloud computing mode to realize an efficient interconnection and information sharing between vehicles, road edge nodes and cloud center. For this example, the vehicle can integrate sensors such as LiDAR and cameras, then share the collected data with road edge nodes and surrounding vehicles for extending the sensing capabilities while enabling vehicle-to-vehicle and vehicle-to-road collaboration. Besides, the cloud center is responsible for collecting data from the widely distributed edge nodes, sensing from the operating conditions of the traffic system and issuing reasonable scheduling instructions for the edge nodes, traffic signal systems and vehicles through intelligent algorithms, thereby improving the operating efficiency

Authors' addresses: Yishan Chen, chenys@jxust.edu.cn, JiangXi University of Science and Technology, Ganzhou, China, 341000; Wei Li, liwei@jxust.edu.cn, JiangXi University of Science and Technology, Ganzhou, China, 341000; Junhong Huang, drhughhjh@jxust.edu.cn, JiangXi University of Science and Technology, Ganzhou, China, 341000; Honghao Gao*, gaohonghao@shu.edu.cn, Shanghai University, Shanghai, China, 200444; Shuiguang Deng*, dengsg@zju.edu.cn, Zhejiang University, Hangzhou, China, 310058.

ACM 1550-4859/2024/3-ART
https://doi.org/10.1145/3652515

of the traffic system and minimizing road congestion. However, in the above scenario, vehicles have data privacy concerns when in conjunction with cloud-edge collaboration and there is no highly flexible and secure computation offloading model for information exchange at local-edge-cloud. Therefore, we need to come up with a three-layer local-edge-cloud orchestrated architecture (Figure 1) which can minimize the operational time for latency-sensitive tasks requested by vehicles that have data privacy concerns, while each task can be executed under local, edge or cloud computing modes with no need to rely on privacy data [39].

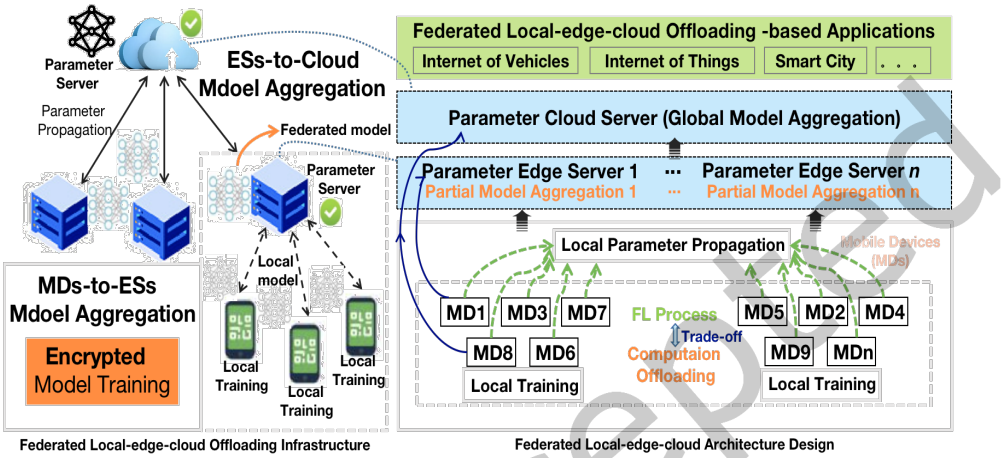


Fig. 1. Three-layer Local-edge-cloud Orchestrated System Architecture.

Existing works usually use game theory, heuristic algorithms or machine learning to obtain optimal the local-edge-cloud offloading decision. Yet, decisions made by them are often time undetermined or privacy leaked. Xu et al. [30] proposed a two-stage computing offloading algorithm which discussed the energy consumption under a time-delay constraint. This approach can not guarantee the timeliness and security of the collaboration. Xue et al. [32] designed a novel optimization method for parallel offloading of large-scale DNN models in a local-edge-cloud collaborative environment with limited resources, which ensured a low "delay-energy-cost" coordination but resulted in a privacy concern when transmitting the data. Moreover, previous studies never investigated the time optimization and privacy protection problem as a whole in local-edge-cloud optimal offloading; most of the existing works individually tackled the time minimization problem without privacy concerns. Therefore, a better local-edge-cloud optimal offloading strategy for latency and privacy sensitive tasks is required.

To exploit a multi-computation power of mobile devices (MDs, e.g., vehicles), edge servers (ESs, e.g., road edge nodes) and Cloud, a partial computation offloading is necessary. To this aim, several factors should be considered such as signal coverage, MDs' mobility and location, server processing capacity, Quality of Services (QoS) level, etc, for the following reasons. On one hand, MDs with different performing capabilities always lack in handling computation-intensive and latency-sensitive services, coupled with the tasks' frequent interruption due to the signal coverage difference raised by the MDs' changing location, which leads to the wasted computation resources and energy, then further increases the task computation offloading process's complexity. On the other hand, the complex network environment formed by the distributed deployment of MDs, ESs and Cloud may lead to data snooping when offloading, and the ESs or Cloud toward which the task is offloaded may also have malicious behaviors. Therefore, it is crucial to rationally optimize the multi-computation power allocation among heterogeneous MDs, ESs and Cloud, to enhance the offloading model's resilience, and to serve a privacy protection for latency-sensitive tasks.

In this paper, we present a local-edge-cloud optimal offloading strategy for latency and privacy sensitive tasks based on Federated Learning (FL) method with Differential Evolution (DE) algorithm. First, Local, Edge, Cloud and Federated Models are built to obtain the latency incurred under different computing modes. Second, we consider a three-layer hierarchical network architecture composed of MDs, ESs and Cloud. The strategy aims at performing task offloading within a FL architecture supported by a hierarchical federated averaging (HierFAVG) algorithm that allows multiple ESs and the Cloud to perform partial and global model aggregation respectively. Since FL process takes time, hence impacting on the time resources left to the MDs' willing to offload, which in turn affects the model aggregations as represented by parameter propagation among MDs, ESs and the Cloud. Therefore, the FL architecture enables learning information that is useful for the computation offloading decision phase, and we aim at optimizing the latency allocation between FL and offloading phases by DE algorithm.

Our contribution is threefold:

- We investigate a latency and privacy sensitive task offloading strategy in a hierarchical federated local-edge-cloud architecture, supported with a HierFAVG algorithm that performs partial and global model aggregation in MDs-to-ESs and ESs-to-Cloud collaboration.
- We model the joint FL and offloading process through its latency consumption, and then formulate the latency minimization as a constrained non-linear optimization problem solved by DE algorithm. Based on the DE algorithm, the computation offloading parameters and the optimal number of FL process iterations can be well balanced to achieve the goal of latency minimization.
- We conduct simulations on the local-edge-cloud offloading strategy and confirm the high performance of FL architecture with DE.

The organization of this paper is as follows. We survey state of the art in Section II. In Section III, the system model and its related four sub-models will be introduced. After that, the optimization problem to be solved is detailed and explained. In Section IV, DE-based solution is thoroughly conducted. The simulation results and conclusion of this paper will be given in Section V and Section VI, respectively.

2 RELATED WORK

With the increasing demand for computing power in the industry, more and more computing platforms are combining cloud computing, edge computing and local computing to accelerate the deep integration of quality services [12, 15, 35]. This section introduces the latest domestic and international researches from two aspects: Federated Collaboration and Task Offloading.

Federated Collaboration. 1) From the resource collaboration's perspective, Guo et al. introduced a concept of hybrid fiber-optic wireless (FiWi) networks where multiple access edges and centralized cloud collaborated in concert to provide better offloading performance and good scalability as the number of computing tasks increased. The combination of edge computing and cloud computing fully utilized the advantage of edge computing's scalability, high mobility, multiple wireless access technologies and cloud computing's high capacity, high reliability and low latency [9]; Li et al. adopted a collaborative cloud-edge resource scheduling approach to make up for the lack of resources at the edge. Resource scheduling from the cloud data centre to the memory or disk of each edge server could provide the resources required for tasks at the edge, which improved the cloud resource utilization to some extent [11]; For resource-intensive applications such as big data analysis, AI processing and 3D sensing in IIoT devices, Hong et al. [10] proposed a multi-hop cloud-edge collaborative computing offload paradigm that aiming to minimize transmission and computation time in task offloading process [27]; Dinh et al. designed a computing resource renting strategy for cloud-edge to expand the capacity of the edge nodes with the goal of minimizing the total cost, including the processing cost at the edge, the cost of remote on-demand virtual machines, and the cost of reserving remote reserved virtual machines [6].

2) From the management collaboration's perspective, Deng et al. formulated the application deployment problem into a problem of deploying heterogeneous microservice instances, while modeling servers as queuing nodes. They set the computation and storage costs of microservices as constraints to achieve the goal of reducing deployment costs while satisfying the average response time constraint of mobile services [5]; Ozcan et al. focused on the system fault tolerance by creating a Docker with partitioned placement of a executable application's secure copy, which saved the development time and provided a new idea for the Industrial Internet to debug applications remotely at edge devices [19]; Sampaio et al. proposed an adaptation mechanism for automatically managing the microservices' placement in applications based on the correlation between microservices and resource usage, which could ultimately save up to 80% on host usage, thus reducing resource usage and improving application [21].

3) From the application collaboration's perspective, to handle the amount of services generated by IoT devices in smart cities, Xu et al. proposed a trust-oriented IoT service placement scheme that addressed improvements in resource usage, load balancing and energy consumption while protecting the privacy of IoT devices [31]; Chen et al. proposed an architecture of edge computing for IoT-based manufacturing. It also analyzed the role of edge computing from four aspects including edge equipment, network communication, information fusion, and cooperative mechanism with cloud computing [2]; Wang et al. studied resource management for smart home healthcare systems in a cloud-edge architecture and proposed a task scheduling scheme called Health-Edge that could process different tasks based on priority to reduce latency [26].

Task Offloading. 1) From the computation offloading's perspective, Lyu et al. designed asymptotically optimal schedules tolerant to out-of-date network knowledge, thereby relieving stringent requirements on feedbacks, and their approach was able to dramatically reduce feedbacks at no cost of optimality [18]; Chen et al. studied the multi-user computation offloading problem for mobile-edge cloud computing in a multi-channel wireless interference environment. They showed that it was NP-hard to compute a centralized optimal solution, and hence adopted a game theoretic approach for achieving efficient computation offloading in a distributed manner. Meanwhile, they designed a distributed computation offloading algorithm that could achieve a Nash equilibrium over the centralized optimal solutions [3]; Wei et al. purposefully postponed the offloading tasks by actively intervening in the splittable tasks, while taking the task execution process into account on the MEC as well as the transmission process. Such an offloading method could ensure that multiple tasks' non-overlapping time occupation on MEC servers, avoiding the latency caused by tasks waiting in line and efficiently utilizing the computational resources of the MEC [29].

2) From the security's perspective, Fredrikson et al. developed a model inversion algorithm that compared the target feature vector with each possible value and then obtained a weighted probability estimation of the correct value. Experimental results showed that the algorithm was very effective in exploiting information from decision tree or face recognition based training models [7]; To address the problem of easy exposure for clients' privacy, Geyer et al. proposed a client-side differential privacy-preserving federal optimization algorithm. Hiding clients' contributions during training could maintain an equilibrium between privacy loss and model performance [8]; Yu et al. proposed a Federated Learning Incentivizer (FLI) payoff-sharing scheme. The scheme dynamically divided a given budget in a context-aware manner among data owners in a federation by jointly maximizing the collective utility while minimizing the inequality among the data owners, in terms of the payoff gained by them and the waiting time for receiving payoff [34]. Xu et al. proposed a two-stage computing offloading algorithm which was divided into two stages, and both stages were designed to offload algorithms based on game theory. Through game theory, the authors first obtained the edge-to-local optimal offloading decision, then found the cloud-edge optimal offloading decisions for all local devices that chose to offload to the edge, and finally reached the goal of optimal offloading on the three sides of the local-edge-cloud [30].

3) From the fault tolerance's perspective, Tuli et al. proposed a composite AI model (PreGAN) using a Generative Adversarial Network (GAN) to predict preemptive migration decisions for proactive fault-tolerance

in containerized edge deployments. PreGAN used co-simulations in tandem with a GAN to learn a few-shot anomaly classifier and proactively predicted migration decisions for reliable computing [25]; Xue et al. proposed an energy-aware fault-tolerant resource scheduling algorithm to improve system reliability while minimizing the energy consumption. They allocated resources by reliability and energy-aware resource scheduling method for tasks firstly. Then, CPU temperature prediction and time between failures (TBF) prediction were used to trigger proactive fault tolerance mechanism (VM migration) [33].

In several of these works, authors aimed at optimizing the collaborative offloading process without paying much attention to latency and privacy requirements. Also, hierarchical federated local-edge-cloud model can have a significant contribution on high-quality offloading and is required to be explored. Therefore we aim to perform an operational time optimization of latency sensitive tasks in local-edge-cloud collaboration scenarios by considering the latency constraints while addressing the privacy concerns.

3 SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we focus on presenting a system model for local-edge-cloud orchestrated computing, followed by local model, edge model, cloud model and federated model. Here we list the necessary notations used in this paper, as shown in Table 1.

We consider a three-layer local-edge-cloud real life scenario shown in Figure 2, containing of one cloud server, a set $\mathcal{D} = \{d_1, \cdots, d_m, \cdots, d_M\}$ of M MDs and a set $\mathcal{E} = \{e_1, \cdots, e_n, \cdots, e_N\}$ of N ESs with Random Walk from Uniform Distribution [13].

The generic mth MD, whose movement trajectory is arbitrary, is defined by a processing capability per CPU cycle equal to $\tau_{d,m}$ Floating Point Operations per Second (FLOPS) with a CPU frequency of $f_{d,m}$, while its available memory is D_m. Each MD is supposed to be able to achieve a terrestrial communication bandwidth of $B^E_{d,m}$ with ESs, which enables a stable data transmission. On the other hand, it should communicate with a $B^C_{d,m}$ bandwidth, while communicating with the Cloud in a non-terrestrial communication network. During the computation offloading process of local-edge-cloud collaboration, including MD computation, MD-ES communication, MD-Cloud communication, ES computation, ES-MD communication, ES-Cloud communication, FL computation, FL communication, etc., come with strict latency requirements. Such latency requirements need to be considered when solving this computational offloading job. Thus, the mth MD's task information is denoted as x_m, which is identified by the tuple $\langle D_{x_m}, \Omega_{x_m}, \bar{T}_{x_m} \rangle$, where D_{x_m} is the task information size in Bytes, Ω_{x_m} is the required CPU execution cycles, and $\bar{T}_{x_m}$ is the maximum allowed latency.

The generic nth ES is characterized by processing capability equal to $\tau_{e,n}$ Floating Point Operations per Second (FLOPS) per CPU cycle, and its CPU frequency is $f_{e,n}$. On one hand, each ES should be able to communicate with a bandwidth of $B^D_{e,n}$ and cover a service area of radius $R_{e,n}$. In addition, it provides computation offloading services to the MDs within its connecting area. On the other hand, the area should be within the coverage of the cloud server, which has more computing power compared to ESs or MDs. For the coverage area, we place the Cloud at a height h_C above the ground in the simulation environment.

3.1 Local Model

Once a task x_m is generated on the mth MD at time slot t, the mth MD should immediately decide to execute the task locally or offload it to the ES or the Cloud within their connecting area. We assume that the mth MD is able to split the task x_m in three portions and use a three-valued variable $(\alpha_m, \beta_m, \delta_m)$ to indicate the portions offloaded to any of the ESs or the Cloud, Whereas, $\alpha_m + \beta_m + \delta_m = 1$. Specifically, the portion α_m can be locally computed, the portion β_m can be offloaded to the nearby ESs, the portion δ_m can be offloaded to the Cloud. Our objective is to obtain the optimal offloading decision $(\alpha^*_m, \beta^*_m, \delta^*_m)$ by balancing the FL and offloading process which will be discussed later.

Table 1. Notation Summary

Notation	Description
$\mathcal{D}$	M Mobile Devices
$\mathcal{E}$	N Edge Servers
x_m	Task Generated by the mth MD
$\tau_{d,m}$	Processing Capability of the mth MD
$f_{d,m}$	CPU Frequency of the mth MD
D_m	Available Memory of the mth MD
$B^E_{d,m}$	Allocated Bandwidth for the mth MD Communicating with ESs
$B^C_{d,m}$	Allocated Bandwidth for the mth MD Communicating with the Cloud
$B^D_{e,n}$	Allocated Bandwidth for the nth ES Communicating with MDs
D_{x_m}	Task Information Size of the Task x_m
Ω_{x_m}	Requested CPU Cycles of the Task x_m
$\bar{T}_{x_m}$	Maximum Latency Allowed of the Task x_m
$\tau_{e,n}$	Processing Capability of the nth ES
$f_{e,n}$	CPU Frequency of the nth ES
D_n	Available Memory of the nth ES
$R_{e,n}$	nth ES's Connecting Area
h_C	The Cloud Altitude from the Ground
$\alpha_m, \beta_m, \delta_m$	Offloading Decision
$T^{soj}_{m,n}$	Sojourn Time Available for the mth MD before Leaving the nth ES
$T^{soj}_{m,C}$	Sojourn Time Available for the mth MD before Leaving the Cloud
T^{loc}_m	Local Computation Time
$T^{M/E,tx}_{m,n}$	MD-ES Transmission Time
$T^{M/C,tx}_{m,C}$	MD-Cloud Transmission Time
$T^{x_m}_n$	Edge Processing Time
$T^{E/M,rx}_{n,m}$	ES-MD Results Backing Time
$T^{x_m}_C$	Cloud Processing Time
$T^{C/M,rx}_{C,m}$	Cloud-MD Results Backing Time
$T^{FL,cx}_m$	FL Processing Time
$T^{FL,prop}_{m,it}$	FL Propagation Time
$T^{FL,tx}_{m,k_1}$	MD-ES Local Parameter Transmission Time
$T^{FL,tx}_{n,k_1k_2}$	ES-Cloud Partial Parameter Transmission Time
$T^{FL,rx}_{m,k_1k_2}$	Cloud-ES Global Parameter Transmission Time
$T^{FL}_{mnC,it}$	Single FL Iteration Time in Local-edge-cloud Collaboration

It is worth to be noted that the MDs can be on motion or static, while the ESs or the Cloud are always static and their locations are fixed. Therefore, to offer a trouble-free computation offloading within the availability of computing resources, we now investigate the relative position information based on the MDs, ESs and the Cloud. We suppose that the generic mth MD is located at position $\{x_{d,m}(t), y_{d,m}(t)\}$ in time slot t, while the generic nth ES and the Cloud are considered to be in a fixed position $\{x_{e,n}, y_{e,n}\}$, $\{x_c, y_c\}$, respectively. Hereafter, We now

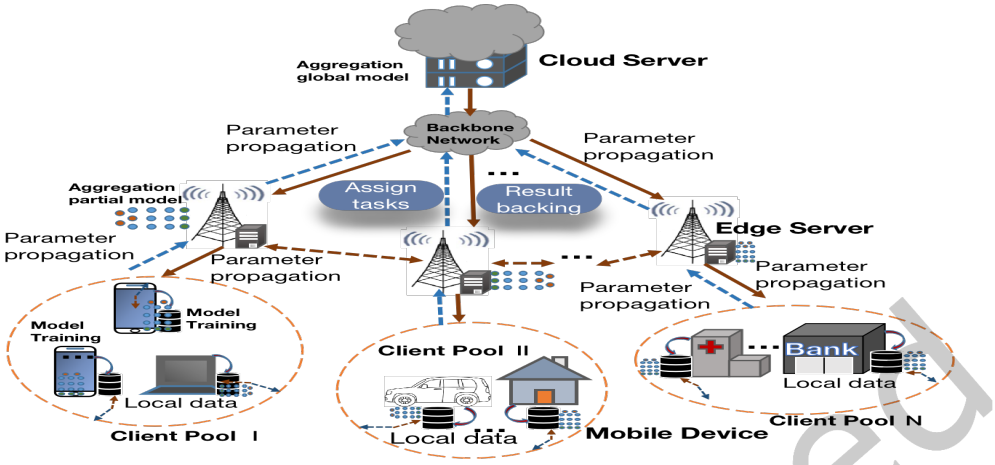


Fig. 2. Scenario.

discuss separately the collaboration time available within the connecting area of one ES and the Cloud during the movement of the mth MD. It has to be clarified that in this work we just discuss the movement of a MD within one ES's connecting area, without considering overlapping problems of multiple ESs.

Hence, the remaining distance within which the mth MD remains under the connecting area of the nth ES can be defined as:

$$\prod_{m,n} = cos\theta\sqrt{x_{d,m}^2(t) + y_{d,m}^2(t)} \pm \sqrt{(x_{d,m}^2(t) + y_{d,m}^2(t))(cos^2\theta - 1) + R_{e,n}^2} \tag{1}$$

where $R_{e,n}$ is the coverage radius of the nth ES's connecting area; θ is the direction angle of the three-point based on the mth MD's starting point, the nth ES's centre point and the point where the MD locates in the edge of the mth ES's connecting area.

Then, the sojourn time, the duration available for the mth MD before leaving the nth ES's connecting area, can be calculated as:

$$T_{m,n}^{soj} = \frac{\prod_{m,n}}{v_m} \tag{2}$$

where v_m is the movement speed of the mth MD.

Similar to the nth ES, the remaining distance within which the mth MD remains under the Cloud's connecting area can be defined as:

$$\prod_{m,C} = cos\theta\sqrt{x_{d,m}^2(t) + y_{d,m}^2(t)} \pm \sqrt{(x_{d,m}^2(t) + y_{d,m}^2(t))(cos^2\theta - 1) + R_C^2} \tag{3}$$

where R_C is the coverage radius of the cloud server's connecting area. It is worth to be noted that the formula (3) is valid only if the mth MD locates in the connecting area of the Cloud. Then, the sojourn time for the mth MD before leaving the Cloud's connecting area can be calculated as:

$$T_{m,C}^{soj} = \frac{\prod_{m,C}}{v_m}. \tag{4}$$

MD local computation. After the mth MD generates a task, the MD can choose to locally compute all or part of this task based on its limited computing and storage resources. Hence, the local computation time is represented as:

$$T_m^{loc} = \alpha_m \cdot \frac{\Omega_{x_m}}{\tau_{d,m} f_{d,m}} \tag{5}$$

where α_m is the portion of the task scheduled to be computed on the mth MD locally.

MD-ES communication. Considering a portion β_m of the task needs to be offloading to the nth ES. First, the mth MD need transmit a task data with a memory size of $\beta_m D_{x_m}$ to the nth ES, and therefore certain time is consumed.

In the following, we employ the widely-used communication model of radio access, to calculate the transmission rate between the mth MD and the nth ES, defined as:

$$r_{m,n}(B_{d,m}^E, s_{m,n}) = B_{d,m}^E log_2(1 + \frac{P_m^{tx} \cdot h(s_{m,n})}{N_0 B_{d,m}^E}) \tag{6}$$

where P_m^{tx} is the transmission power of the mth MD, $h(s_{m,n})$ is the corresponding channel power gain at a distance $s_{m,n}$ between the mth MD and the nth ES, N_0 is the noise power spectral density, $B_{d,m}^E$ is the allocated bandwidth associated to the mth MD during the communication with ESs.

Hence, the MD-ES transmission time required for the portion β_m of the task from the mth MD to the nth ES can be given as:

$$T_{m,n}^{M/E,tx} = \frac{\beta_m D_{x_m}}{r_{m,n}(B_{d,m}^E, s_{m,n})} \tag{7}$$

where $r_{m,n}(B_{d,m}^E, s_{m,n})$ is the uplink data rate from the mth MD towards the nth ES, which depends on the allocated bandwidth resources and the MD-ES's distance.

MD-Cloud communication. Similar to the MD-ES transmission time, the MD-Cloud transmission time can be defined as:

$$T_{m,C}^{M/C,tx} = \frac{\delta_m D_{x_m}}{r_{m,C}(B_{d,m}^C, s_{m,C})} \tag{8}$$

where $r_{m,C}(B_{d,m}^C, s_{m,C})$ is the uplink data rate from the mth MD towards the Cloud.

3.2 Edge Model

Considering the portion β_m of the task x_m generated on the mth MD is offloaded to the nth ES. Then, the next two operations on ESs need be completed, which consists of Edge processing and ES-MD results backing.

Edge processing. After the portion β_m of the task x_m transmits to the nth ES, the computation on the nth ES starts. Hence, the edge processing time can be represented as:

$$T_n^{x_m} = \beta_m \cdot \frac{\Omega_{x_m}}{\tau_{e,n} f_{e,n}} \tag{9}$$

where $\tau_{e,n}$ is the nth ES's processing capability while $f_{e,n}$ is the CPU frequency of the nth ES.

ES-MD results backing. After the completion of the nth ES computation, the ES needs to send back the execution result to the mth MD. The time required by the mth MD for receiving the result from the nth ES is given as:

$$T_{n,m}^{E/M,rx} = \frac{\beta_m D_{x_m,re}}{r_{n,m}(B_{e,m}^D, s_{n,m})} \tag{10}$$

where $r_{n,m}(B_{e,m}^D, s_{n,m})$ is the downlink data rate from the nth ES towards the mth MD; $D_{x_m,re}$ is the task processing result at the nth ES.

3.3 Cloud Model

Comparing to resource-constrained MDs and ESs, the Cloud has strong computing power and enough storage resources. However, latency-sensitive task processing on Cloud is not necessarily a best choice due to its excessive communication overhead and long latency. Therefore, in this paper, we give a priority to the MD local computing and edge processing, while the cloud processing will only be considered if there are high computing power required services. Hence, similar to the edge model, the cloud operation also contains two steps: Cloud processing and Cloud-MD results backing as follows.

Cloud processing. After the portion δ_m of the task x_m transmits to the cloud, the Cloud begins to execute, and the Cloud processing time can be represented as:

$$T_C^{x_m} = \delta_m \cdot \frac{\Omega_{x_m}}{\tau_C f_C} \tag{11}$$

where τ_C is the cloud server's processing capability while f_C is the CPU frequency of the Cloud.

Cloud-MD results backing. After the completion of the Cloud processing, the Cloud needs to send back the processing result to the mth MD. The time required by the mth MD for receiving the result from the Cloud is given as:

$$T_{C,m}^{C/M,rx} = \frac{\delta_m D_{x_m,rc}}{r_{C,m}(B_{C,m}^d, s_{C,m})} \tag{12}$$

where $r_{C,m}(B_{C,m}^d, s_{C,m})$ is the downlink data rate from the Cloud towards the mth MD; $D_{x_m,rc}$ is the task processing result at the Cloud.

3.4 Federated Model

We exploit the privacy properties in the federated learning framework [36, 40]. The framework has a high security, which ensures that local-edge-cloud parties build models jointly with cloud servers, edge servers, and local devices without disclosing the underlying data and the encrypted form of the underlying data, by exchanging the model parameters instead of the raw data set in tasks and the data in nodes that needs to be protected. There are two roles in a traditional FL model: FL server and FL clients. In a general federated learning process, FL server assists distributed FL clients to complete the model training by exchanging parameters. From the Cloud's perspective, it can access more data and provide services for more clients, but it has excessive communication overhead and long latency. Comparing to the Cloud, ESs are allowed for more efficient communication with clients, however, ESs have limited resources and can not support enough clients. So as to combine their strengths, in this paper, we propose a local-edge-cloud hierarchical federated learning model, which can accommodate both the Cloud's global model aggregation and multiple ESs' partial model aggregation. In this way, models can be trained faster and a better communication-computation trade-off can be achieved.

Compared to Cloud-based FL, hierarchical FL will significantly reduce the communication cost with the Cloud, complemented by efficient client-edge updates, resulting in a significant reduction in runtime and the number of local iterations. On the other hand, hierarchical FL will outperform Edge-based FL in model training due to the greater amount of data accessible to the Cloud. Generally, hierarchical FL contains two steps: 1)

Algorithm 1: hierarchical federated averaging (HierFAVG)

Input: Initialized all MDs' learning model with parameter $\mathbf{w}_0$
Output: Optimal parameter $\mathbf{w}_m^n(k)$
for $k = 1, 2, \cdots, K$ **do**
  for *each MD* $m = 1, 2, \cdots, M$ *in parallel* **do**
    $\mathbf{w}_m^n(k) \leftarrow \mathbf{w}_m^n(k-1) - \eta \bigtriangledown F_i(\mathbf{w}_m^n(k-1))$
  end
  if $k \mid k_1 = 0$ **then**
    for *each ES* $n = 1, 2, \cdots, N$ *in parallel* **do**
      $\mathbf{w}^n(k) \leftarrow EdgeAggregation(\mathbf{w}_m^n(k)_{m \in M^n})$
      if $k \mid k_1 k_2 \neq 0$ **then**
        for *each MD* $m \in M^n$ *in parallel* **do**
          $\mathbf{w}_m^n(k) \leftarrow \mathbf{w}^n(k)$
        end
      end
    end
  end
  if $k \mid k_1 k_2 = 0$ **then**
    $\mathbf{w}(k) \leftarrow CloudAggregation(\mathbf{w}^n(k)_{n=1}^N)$
    for *each MD* $m = 1, 2, \cdots, M$ *in parallel* **do**
      $\mathbf{w}_m^n(k) \leftarrow \mathbf{w}(k)$
    end
  end
end

MD-ES partial model aggregation; 2) ES-Cloud global model aggregation. We assume that the Cloud acts as the FL server in ES-Cloud global aggregation, while ESs acts as the FL severs and MDs under the related ESs' connecting area act as FL clients in MD-ES partial model aggregation. In other words, we consider a model aggregation of three hierarchical layers: MDs, ESs and the Cloud. In the following, firstly, we introduce the general learning problem of FL, because Cloud-based FL systems and Edge-based FL systems differ only in terms of communication and the number of clients involved, and they are architecturally identical. Therefore, we treat them as the same traditional two-hierarchical FL system in this section and introduce a three-layer optimization algorithm-HierFAVG [17, 20, 24].

3.4.1 Federated learning. HierFAVG shown in the Algorithm (1) is executed in a federated way among all the involved nodes (Cloud, ESs, MDs) through parameter exchange, and the parameters are the weights of the implemented HierFAVG algorithm. Firstly, initializing the learning model over all the involved nodes with the parameter $\mathbf{w}_m^n(k)$. Secondly, training the local models (MDs), then obtain the partial losses and global losses through weighted average calculation of local losses and partial losses, respectively. Thirdly, exchanging parameters over wireless links toward MD-ES and ES-Cloud. Finally, collecting parameters and aggregating models from the Cloud, ESs, MDs, which means that the optimal model parameter $\mathbf{w}_m^n(k)$ can be obtained.

(1) FL Training: In this work, we focus on an unsupervised federal learning, and FL computation contains the local/partial training of the machine learning model based on MDs/ESs dataset. Since the local training and partial training are architecturally identical, we only discuss the local training in the following.

The training data set is supposed to be denoted as $D = \{a_j, b_j\}$ and distributed on M MDs, where $|D|$ represents the total training samples. We denote a_j as the jth input sample, while b_j is the corresponding label. Then, the loss function of the jth data sample can be denoted as $f(a_j, b_j, w)$, which represents the prediction error of the model with respect to the jth data sample. Hence, the partial loss function of the total training dataset can be given as:

$$F(w) = \frac{1}{|D|} \sum_{j=1}^{|D|} f(a_j, b_j, w) = \frac{1}{|D|} \sum_{j=1}^{|D|} f_j(w) \tag{13}$$

where w is the parameter of the machine learning model. Since D cannot be accessed directly by the parameter server (ESs or Cloud). Therefore, by training the dataset, we can calculate the parameters that will be propagated over the nodes (MDs or ESs). Based on the gradient descent method, we have:

$$w(k) = w(k-1) - \eta \bigtriangledown F(w(k-1)) \tag{14}$$

where k is the index of the updating, η is a step size of the gradient descent.

Specifically, the partial loss function $F(w)$ can be gained from weighted average of local loss functions over D_i, instead of being calculated directly, which can be given as:

$$F(w) = \frac{\sum_{i=1}^{M} |D_i| F_i(w)}{|D|} \tag{15}$$

$$F_i(w) = \frac{\sum_{j \in D_i} f_j(w)}{|D_i|}. \tag{16}$$

Furthermore, we need to explore more about the parameter $w(k)$. After each k_1 local updating on the mth MD, each ES aggregates the MDs models within its connecting area. Then after each k_2 ES model aggregations, the Cloud aggregates all the ESs models, which means that the MDs' communication with the Cloud happens every $k_1 k_2$ local updates. During local updates, the mth MD has to compute the local parameter $w_{d,m}^{it}$ based on $|D|_m$ data samples, while $it (it = k_1 k_2)$ represents the iteration number in training. For each iteration, the total number of FLOPs required for each data sample z is ϕ_z, the time consumed during FL process at the mth device is given as:

$$T_m^{FL,cx} = \frac{\sum_{z=1}^{|D|_m} \phi_z}{\tau_{d,m} f_{d,m}}. \tag{17}$$

(2) FL Communication: After the FL training, MDs/ESs communicate the local model updates towards the ESs/Cloud in uplink and receive back the updated partial/global model parameters in downlink. Both uplink and downlink communication processes are characterized by transmission and propagation delays, due to the high distance between MDs/ESs and ESs/Cloud. The propagation time required for each FL iteration is no more than:

$$T_{m,it}^{FL,prop} = 2 \cdot \frac{s_{m,C}}{\gamma}, \forall m \tag{18}$$

where γ is the propagation speed in the considered transmission medium; $s_{m,C}$ is the distance between the mth MD and the Cloud, which can be calculated by using the Cloud altitude (h_C) and the mth MD location through simple algebraic passages, and the multiplication by 2 is due to the two-way propagation delay [23].

For each k_1 local updates, the mth MD needs to send the parameter set $w_{d,m}^{k_1}$ to the ESs for partial model aggregation, while $|w_{d,m}^{k_1}|$ is supposed to represent the data size of the local parameter expressed in bits, the uplink transmission time required for the FL parameters in local updates k_1 can be given as:

$$T_{m,k_1}^{FL,tx} = \frac{|w_{d,m}^{k_1}|}{r_{m,E}(B_{d,m}^{E}, s_{m,E})} \tag{19}$$

where, $r_{m,E}(B_{d,m}^{E}, s_{m,E})$ is the uplink transmission rate between mth MD and the ESs during k_1 local updates, which is a function of the MDs bandwidth ($B_{d,m}^{E}$), and the distance ($s_{m,E}$) between the mth MD and the ESs.

Accordingly, for each k_1k_2 local updates, the Cloud performs the global aggregation of the received partial model parameters from ESs to create a global parameter vector $w_C^{k_1k_2}$ for the next iteration it and transmits it back towards MDs over the downlink communication links. Therefore, the uplink transmission time for the partial parameter $w_{n,C}^{k_1k_2}$ is given as:

$$T_{n,k_1k_2}^{FL,tx} = \frac{|w_{n,C}^{k_1k_2}|}{r_{n,C}(B_n^{C}, s_{n,C})} \tag{20}$$

where, $r_{n,C}$ is the uplink transmission rate between the nth ES and the Cloud during k_1k_2 local updates, which is a function of the nth ES bandwidth (B_n^{C}), and the distance ($s_{n,C}$) between the nth ES and the Cloud. Then, in downlink, the global parameter $w_C^{k_1k_2}$ transmission time is:

$$T_{m,k_1k_2}^{FL,rx} = \frac{|w_C^{k_1k_2}|}{r_{C,m}(B_C^{D}, s_{C,m})} \tag{21}$$

Hence, the total time required for a single FL iteration can be given as:

$$\begin{aligned} T_{mnC,it}^{FL} &= T_m^{FL,cx} + T_{m,it}^{FL,prop} + \max_m\{T_{m,k_1}^{FL,tx}\} + \max_n\{T_{n,k_1k_2}^{FL,tx}\} + T_{m,k_1k_2}^{FL,rx} \\ &= T_m^{FL,cx} + T_{m,it}^{FL,prop} + \max_m\{T_{m,\frac{it}{k2}}^{FL,tx}\} + \max_n\{T_{n,it}^{FL,tx}\} + T_{m,it}^{FL,rx}. \end{aligned} \tag{22}$$

3.4.2 Federated offloading. The federated offloading problem focuses on setting the offloading decision $\alpha_m, \beta_m, \delta_m$ in an optimal way such that service latency ($\overline{T}_{x_m}$), sojourn time ($T_{m,n}^{soj}$)/($T_{m,C}^{soj}$) are respected. Therefore, the latency constrained optimization problem can be solved by finding the optimal three-valued variable $\mathbb{V} = \{\alpha_m^*, \beta_m^*, \delta_m^*\}$ such that:

$$T_m^{x_m}(\alpha_m, \beta_m, \delta_m) = \max\{T_{m,C}^{M/C,tx} + T_C^{x_m} + T_{C,m}^{C/M,rx}, T_{m,n}^{M/E,tx} + T_n^{x_m} + T_{n,m}^{E/M,rx}, T_m^{loc}\} \tag{23}$$

where we suppose that offloading and local computation can be performed in parallel, and then:

$$\mathbb{V}^* = \text{argmin}_{\mathbb{V}} \frac{1}{M} \sum_{m=1}^{M} T_m^{x_m}(\alpha_m, \beta_m, \delta_m) \tag{24}$$

subject to the following constraints while combining with the previous definitions given in the system model:

$$T_m^{x_m}(\alpha_m, \beta_m, \delta_m) \leq \overline{T}_{x_m}, \forall m \tag{25a}$$

$$T_{m,C}^{M/C,tx} \leq T_{m,C}^{soj}, \forall m \tag{25b}$$

$$T_{m,n}^{M/E,tx} \leq T_{m,n}^{soj}, \forall m, \forall n \tag{25c}$$

$$\sum_{n=1}^{N} p_E(m,n) \leq 1, \forall m \in M \tag{25d}$$

$$p_C(m) \in \{0,1\} \tag{25e}$$

$$\sum_{m=1}^{M} p_E(m,n) \cdot B_{d,m}^E \leq B_{e,n}, \forall n \in N \tag{25f}$$

$$\sum_{m=1}^{M} p_C(m) \cdot B_{d,m}^C \leq B_C$$

$$0 \leq \alpha_m \leq 1, 0 \leq \beta_m \leq 1, 0 \leq \delta_m \leq 1, \forall m \in M \tag{25g}$$

$$\alpha D_{x_m} \leq D_m \tag{25h}$$

$$\beta D_{x_m} \leq D_n$$

where (25a) represents that each MD's total task processing time is supposed to be limited by the task latency requirement $\overline{T}_{x_m}$ and (25b/25c) represents that each MD is supposed to be able to complete the computation offloading process before leaving the ESs' or Cloud's connecting area for avoiding additional latency costs. We consider a binary assignment variable $p_E(m,n)$ which is equal to 1 if the mth MD's offloading task is assigned to the nth ES, and 0 otherwise. Similarly, $P_C(m)$ takes the value of 1 if the mth MD's offloading task is assigned to the Cloud, and 0 otherwise. (25d) and (25e) set a constraint that each MD can offload tasks to no more than one ES, while (25f) represents that the bandwidth resources available for all active MDs in a particular ES's connecting area is upper bounded by its bandwidth as that in the Cloud. Eq (25g) limits the offloading parameter value $\alpha_m, \beta_m, \delta_m$ between 0 and 1, while (25h) represents the memory limitations of MDs and ESs.

As mentioned before, MDs need to train over the dataset and calculate the local parameters propagated to the selected ESs, to aim this, several FL iterations required by MDs for reaching a predefined convergence value. Therefore, without loss of generality, we consider that the mth MD will be able to estimate the optimal offloading parameter α_m^{Best} after ρ^{Best} FL iterations, where $\rho^{Best} = \frac{\mathcal{K}}{\overline{M}}$; $\mathcal{K}$ can be considered as a numerical constant setting the overall number of FL iterations required to achieve the convergence, while $\overline{M}$ is the number of MDs participating in the FL training process, so higher the participating MDs, lower the required iterations needed to complete the FL training process. Since each FL iteration consumes a certain amount of computation and communication resources, performing ρ^{Best} iterations over all MDs can be challenging and sometimes might not be feasible given the limited MDs resources and the latency constraints imposed by both service requirements (e.g., D_{x_m}, Ω_{x_m}, $\overline{T}_{x_m}$) and sojourn time ($T_{m,n}^{soj}$)/($T_{m,C}^{soj}$). Therefore, in this work, we consider that the generic mth MD is able to perform up to ρ_m FL iterations while $\rho_m \leq \rho^{Best}$. Then, we define the set $\Gamma = \{\rho_1, \cdots, \rho_m, \cdots, \rho_M\}$ as the number of FL iterations performed by each MD.

Since the FL process is based on multiple iterations for obtaining the global model parameter by exchanging the local and partial model parameters, the total time required for the FL process when focusing on the mth MD

can be written as:

$$T_m^{FL}(\rho_m) = \sum_{it=1}^{\rho_m} T_{m,it}^{FL} \tag{26}$$

where ρ_m is the FL iteration number performed by the mth MD, while $T_{m,it}^{FL}$ is the amount of time spent in each iteration. Meanwhile, the time required for completing both FL iterations and task processing has to be constrained by the maximum service latency requirement $\overline{T}_{x_m}$, which can be given by:

$$T_m(\rho_m, \alpha_m, \beta_m, \delta_m) = T_m^{FL}(\rho_m) + T_m^{x_m}(\alpha_m, \beta_m, \delta_m) \leq \overline{T}_{x_m}. \tag{27}$$

Due to the MDs' mobility, the latencies needed by computation offloading and FL process are also supposed to be bounded by the MDs' sojourn times under ESs' and Cloud's connecting area. Since both the ESs and the Cloud are acting as FL parameter servers, the whole FL phase is supposed to be completed within both the ESs sojourn time and the Cloud sojourn time, hence:

$$T_m^{FL}(\rho_m) \leq T_{m,C}^{soj} \tag{28a}$$

$$T_m^{FL}(\rho_m) \leq T_{m,n}^{soj}. \tag{28b}$$

Moreover, each MD is supposed to be able to finish the offloading process within the ESs sojourn time. Thus:

$$T_m^{FL}(\rho_m) + T_{m,n}^{M/E,tx} + T_n^{x_m} + T_{n,m}^{E/M,rx} \leq T_{m,n}^{soj}. \tag{29}$$

It is worth to be noticed that the overall processing time does not need to be included into the sojourn time's constraint, while only the offloading time should be considered, since the local computation can be also performed out of the ESs' connecting time.

3.5 Problem Formulation

This paper aims at minimizing the total latency by properly setting the offloading decision (parameters: $\alpha_m, \beta_m, \delta_m$) and the FL iteration number ($it = k_1 k_2$) used for determining the offloading parameters. Hence, the problem in (24) can be rewritten as:

$$(\Gamma^*, \mathbb{V}^*) = \text{argmin}_{\Gamma,\mathbb{V}} \left\{ \frac{1}{M} \sum_{m=1}^{M} T_m(\rho_m, \alpha_m, \beta_m, \delta_m) \right\} \tag{30}$$

subject to the constraints (25d)-(25g), (26)-(28), and,

$$\sum_{m=1}^{M} B_{d,m}^{C/E} \leq B_{C/E} \tag{30a}$$

$$0 \leq \rho_m \leq \rho^{Best} \quad \forall d_m \in \mathcal{D} \tag{30b}$$

where the sum of bandwidth resources available for all MDs in non-terrestrial communication links is supposed to be upper bounded by the ESs/Cloud bandwidth resources. Eq (30b) shows an upper bound ρ^{Best} for the FL iteration number required by the mth MD.

Then, the problem defined in (30) can be solved by finding two sets of optimization variables Γ and $\mathbb{V}$, which is hard in obtaining an equilibrium among FL iterations and offloading parameters, since Γ and $\mathbb{V}$ are not two separate variables. As more iterations are performed, higher is the reliability with which the offloading parameter would be estimated through the FL process. Hence, the offloading parameter $\alpha_m, \beta_m, \delta_m$ can be modeled as functions of the FL iteration number, and performed for the purpose of estimating the optimal $\alpha_m^{Best}, \beta_m^{Best}, \delta_m^{Best}$,i.e., $\alpha_m = \alpha_m(\rho_m), \beta_m = \beta_m(\rho_m), \delta_m = \delta_m(\rho_m)$. However, the exact relationship between $\alpha_m/\beta_m/\sigma_m$ and ρ_m is hard to be set due to several factors such as MDs' participating number in the FL process,

servers' connection relationship and communication medium in the FL environment, etc. To the best of our knowledge, there is currently no close-to-reality model in the literature aiming at setting the aforementioned relationship, since many factors in real scenarios are difficult to separate. In this work, without loss of generality, we model the estimated β_m and δ_m as two stochastic values which are distributed by following a truncated normal distribution with mean μ and variance σ^2, where $0 \leq \beta_m \leq 1$, $0 \leq \delta_m \leq 1$. To be noted in particular, the portion α_m of the task x_m is locally computed, therefore, we mainly discuss the distribution of the estimated offloading parameter β_m and δ_m, then, the estimated α_m can be represented as 1-β_m-δ_m. Hence, it is possible to define the probability density function $f_{\beta_m}(\cdot)$, $f_{\delta_m}(\cdot)$ of β_m, δ_m as:

$$f_{\beta_m}(\beta_m;\mu_1,\sigma_1) = \begin{cases} \frac{1}{\sigma_1}\frac{\xi\frac{\beta_m-\mu_1}{\sigma_1}}{\Delta(\frac{1-\mu_1}{\sigma_1}-\frac{-\mu_1}{\sigma_1})} & \text{if } 0 \leq \beta_m \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

$$f_{\delta_m}(\delta_m;\mu_2,\sigma_2) = \begin{cases} \frac{1}{\sigma_2}\frac{\xi\frac{\delta_m-\mu_2}{\sigma_2}}{\Delta(\frac{1-\mu_2}{\sigma_2}-\frac{-\mu_2}{\sigma_2})} & \text{if } 0 \leq \delta_m \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

where $\xi(\cdot)$ and $\Delta(\cdot)$ are the probability density function of the associated standard normal distribution and related cumulative distribution function, respectively, and given as:

$$\xi(x) = \frac{1}{\sqrt{2\pi}}e^{(\frac{-x^2}{2})},\ \Delta(y) = \frac{1+\mathrm{erf}(\frac{y}{\sqrt{2}})}{2}. \tag{33}$$

Take an example of β_m, we assume that the mean value of the distribution μ and the variance σ^2, are related to the ρ_m, FL iterations performed by the mth MD, and ρ_{Best}, through which can estimate the optimal offloading parameter. Then, we have:

$$\mu_1 = \beta_m^{Best}(\rho_m),\ \ {\sigma_1}^2 = (\varepsilon \cdot \frac{\rho^{Best}-\rho_m}{\rho^{Best}})^2 \tag{34}$$

where ε is a numerical constant used to control the distribution model's variance. Since the time spent will be higher when more iterations to be performed in the FL phase, so μ can be also represented a function of the iteration number. According to the aforementioned description, it is clear that we have to find a trade-off between

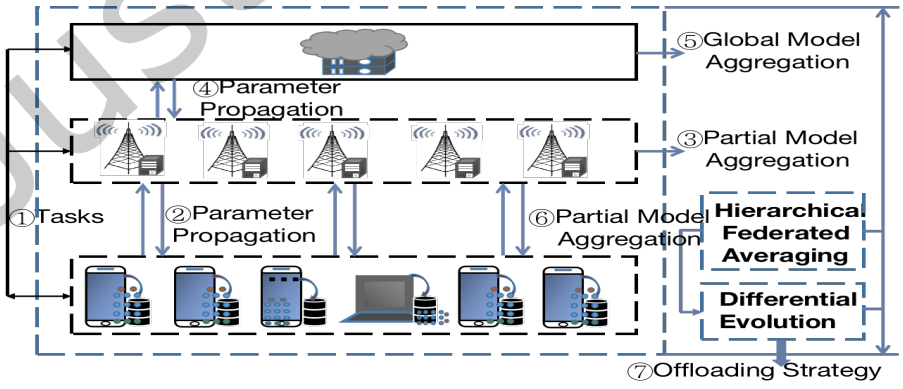


Fig. 3. Workflow Diagram.

offloading and FL processes under the latency constraint. In the following, we need introduce a new parameter $\lambda_m \in [0, 1]$ to model the portion of time allocated for the FL process of the mth MD, and the rest portion with $(1 - \lambda_m) \cdot \overline{T}_{x_m}$ will be allocated for the task processing phase. If $\lambda_m = 0$, the whole time will be allocated for the task processing phase, while if $\lambda_m = 1$, the mth MD will use the whole processing time for the FL phase. Hence, after the tasks are generated by MDs, we consider their target latency as a reference time interval, and the maximum number of possible iterations for the FL process phase can be set as:

$$\rho_m(\lambda_m) \text{ s.t. } T_m^{FL}(\lambda_m) = \sum_{it=1}^{\rho_m(\lambda_m)} T_{m,it}^{FL} \leq \lambda_m \cdot \overline{T}_{x_m} \tag{35}$$

where T_m^{FL} is the total time required for the FL process, and $\overline{T}_{x_m}$ is the maximum service latency; then we consider a set $\Lambda = \{\lambda_1, \cdots, \lambda_m, \cdots, \lambda_M\}$ to describe the time allocated parameters of all MDs. Based on numerous FL iterations, each MD is aiming to find an optimal offloading decision $(\alpha_m^*, \beta_m^*, \delta_m^*)$ to realize the federated local-edge-cloud collaboration by updating the aggregation models without disclosing any data privacy.

Hence, the optimization objective defined in (30) can be rewritten as:

$$\Lambda^* = \text{argmin}_\Lambda \left\{ \frac{1}{M} \sum_{m=1}^{M} T_m(\rho_m(\lambda_m), \beta_m(\lambda_m), \delta_m(\lambda_m)) \right\} \tag{36}$$

subject to the constraints (25d)-(25g), (26), (30), (34) and:

$$0 \leq \lambda_m \leq 1 \quad \forall d_m \in \mathcal{D}. \tag{37}$$

4 SOLUTIONS

Based on the model aggregations in the FL process and DE algorithm described in the following, we can obtain the optimization offloading decision $(\alpha_m^*, \beta_m^*, \delta_m^*)$ and find out appropriate ESs to offload, the workflow FL and DE as described in Figure 3. We assume that the ESs' or Cloud's storage and computing resources are both supposed to be equally shared among all active MDs in their connecting area. As for the MDs managed by the nth ES, the whole of them can be represented as M_n. It has to be clarified that we just need to distinguish the MDs clusters of the nth ES, since there only has one cloud server for tasks needed to be offloaded towards, and we do not need to make decisions. Hence, the solution vector $\Lambda_n = \{\beta_1, \beta_2, \cdots, \beta_{M_n}\}$ can be indexed by M_n values for all MDs connected to the nth ES, and then the problem formulation can be modified as:

$$\Lambda_n^* = \text{argmin}_{\Lambda_n} \left\{ \frac{1}{M_n} \sum_{m=1}^{M_n} T_m(\rho_m(\lambda_m), \beta_m(\lambda_m), \delta_m(\lambda_m)) \right\}. \tag{38}$$

We solve the constrained non-linear optimization problem by DE algorithm. As a population-based heuristic random search algorithm, DE is one of the most effective paradigms of evolutionary algorithms (EAs), originally proposed by Storn and Price in 1995 [4]. Its main idea is to describe the problem solution in the form of a solution vector and the differential value as a random element that creates the variants [14]. Based on the DE algorithm, the computation offloading parameters and the optimal number of FL process iterations can be well balanced to achieve the goal of latency minimization. The DE evolves a population of size N by iterations to obtain the optimal solution, and it generates offspring individuals mainly by three evolutionary operations: mutation, crossover and selection. Assuming that the problem has M (the number of MDs)-dimensional decision variables, the nth target individual of the population in the Gth generation can be expressed as:

$$\overline{X}_{n,G} = [x_{1,n,G}, x_{2,n,G}, x_{3,n,G}, \cdots, x_{M,n,G}] \tag{39}$$

where $n = 1, 2, \cdots, N$ (the number of ESs). When $G = 0$, it is produced by the following equation:

$$\overline{\mathrm{X}}_{\mathrm{m,n,0}} = \mathrm{rand}_{\mathrm{m,n}}[0,1] \cdot (\mathrm{x}_{\mathrm{m,max}} - \mathrm{x}_{\mathrm{m,min}}) \tag{40}$$

where $\mathrm{x}_{\mathrm{m,min}}$ and $\mathrm{x}_{\mathrm{m,max}}$ represent the lower and upper boundary of the mth decision variable $(\alpha_m^*, \beta_m^*, \delta_m^*)$, respectively, and the coefficient $\mathrm{rand}_{\mathrm{m,n}}[0,1]$ is a random number in the range of [0,1].

After initializing the population, the DE iteratively evolves the population using mutation, crossover and selection operations until the algorithm reaches a predefined stopping condition (the maximum number of iterations G_{max} or the maximum number of function evaluations $MaxFEs$). Next, the three operations of the DE in Algorithm (2) are described as follows:

(1) Mutation operation: This operation is a crucial step in the DE, and which is the biggest difference between the DE and other EAs paradigms. The simplest and commonly used mutation strategy "DE/rand/1/bin" is represented by the following formula:

$$\overline{V}_{n,G} = \overline{X}_{r1_n,G} + F \cdot (\overline{X}_{r2_n,G} - \overline{X}_{r3_n,G}) \tag{41}$$

where $\overline{V}_{n,G}$ is the mutation individual corresponding to the target individual $\overline{X}_{n,G}$ produced in the Gth generation. $\overline{X}_{r1_m}, \overline{X}_{r2_m}, \overline{X}_{r3_m}$ are randomly selected from the population with different individuals from each other, and they are all different from the target individual $\overline{X}_{n,G}$. The parameter F is the scaling factor, which is a uniformly distributed random real number in the interval [0,1], and a higher value of F indicates a larger search step, which is favorable for exploration; on the contrary, a lower value of F means that a fine-grained search will be performed around the target individual, which is favorable for exploitation. In addition to "DE/rand/1", the other five widely used DE mutation operators are shown below:

$$\begin{aligned}\mathrm{DE/rand/2/bin} : \mathrm{V_{n,g}} &= \mathrm{X_{r1,g}} + \mathrm{F} \cdot (\mathrm{X_{r2,g}} - \mathrm{X_{r3,g}}) \\ &\quad + \mathrm{F} \cdot (\mathrm{X_{r4,g}} - \mathrm{X_{r5,g}}) \end{aligned} \tag{42a}$$

$$\mathrm{DE/best/1/bin} : \mathrm{V_{n,g}} = \mathrm{X_{best,g}} + \mathrm{F} \cdot (\mathrm{X_{r1,g}} - \mathrm{X_{r2,g}}) \tag{42b}$$

$$\begin{aligned}\mathrm{DE/best/2/bin} : \mathrm{V_{n,g}} &= \mathrm{X_{best,g}} + \mathrm{F} \cdot (\mathrm{X_{r1,g}} - \mathrm{X_{r2,g}}) \\ &\quad + \mathrm{F} \cdot (\mathrm{X_{r3,g}} - \mathrm{X_{r4,g}}) \end{aligned} \tag{42c}$$

$$\begin{aligned}\mathrm{DE/current-to-rand/1/bin} : \mathrm{V_{n,g}} &= \mathrm{X_{n,g}} \\ &\quad + \mathrm{F} \cdot (\mathrm{X_{r1,g}} - \mathrm{X_{n,g}}) + \mathrm{F} \cdot (\mathrm{X_{r2,g}} - \mathrm{X_{r3,g}}) \end{aligned} \tag{42d}$$

$$\begin{aligned}\mathrm{DE/current-to-best/1/bin} : \mathrm{V_{n,g}} &= \mathrm{X_{n,g}} \\ &\quad + \mathrm{F} \cdot (\mathrm{X_{best,g}} - \mathrm{X_{n,g}}) + \mathrm{F} \cdot (\mathrm{X_{r1,g}} - \mathrm{X_{r2,g}}). \end{aligned} \tag{42e}$$

(2) Crossover operation: After generating mutation individuals, this operation is performed for each pair of $\overline{X}_{n,G}$ and $\overline{V}_{n,G}$ to obtain a test individual $\overline{U}_{n,G} = [u_{1,n,G}, u_{2,n,G}, u_{3,n,G}, \cdots, u_{M,n,G}]$. The binomial crossover operation is one of the most common ways of DE, which is represented as follows:

$$u_{n,m}^{(t)} = \begin{cases} v_{n,m}^{(t)}, & \text{if } m = K \text{ or } rand_{n,m}[0,1] \le C_r \\ x_{n,m}^{(t)}, & \text{otherwise} \end{cases} \tag{43}$$

where K is a random integer between 1 and M, $rand_{n,m}[0,1]$ is a random number uniformly distributed in the interval [0,1]. C_r is referred to as the crossover rate, which has a greater impact on DE performance. The larger value of C_r means that the offspring can inherit more information from the mutation individual; on the contrary, the smaller value of C_r, the smaller difference between the offspring individual and the parent individual. It is worth noting that the condition $m = K$ can make the test individual $\overline{U}_{n,G}$ differ from the corresponding target individual $\overline{X}_{n,G}$ in at least one dimension.

Algorithm 2: Differential Evolution Algorithm

Input: Dimension: M, Population: N, Generation: G
Output: Best solution Λ^*
$g \leftarrow 1$ (initialization);
for *i = 1 to N* **do**
  for *j = 1 to M* **do**
    $x_{m,n,g} = x_{m,min} + rand(0,1) \cdot (x_{m,max} - x_{m,min})$;
  end
end
while $(|f(\Lambda^*)| \geq \varepsilon) or (g \leq G))$ **do**
  for *i = 1 to N* **do**
    for *j = 1 to M* **do**
      Mutation and crossover;
    end
    if $f(u^{(g)}) \leq f(x_n^{(g)})$ **then**
      $x_n^{g+1} \leftarrow u^{(g)}$;
      if $f(x_n^{(g)}) < f(\Lambda^*)$ **then**
        $\Lambda^* \leftarrow x_n^{(g)}$;
      end
      else
        $x_n^{g+1} \leftarrow x_n^{g+1}$;
      end
    end
  end
  $g \leftarrow g + 1$;
end
return *Best solution* Λ^*

(3) Selection operation: As the last operation of the DE, the selection operation is a one-to-one competition between the target individual and its corresponding test individual. For the minimization problem, if the objective function value of the test individual is less than or equal to the objective function value of the target individual, the test individual will survive to the next generation; otherwise, the target individual will enter the next generation. This process is in accordance with the natural law of "survival of the fittest", and its mathematical expression is shown below:

$$x_n^{t+1} = \begin{cases} u^{(t)}, & \text{if } f(u^{(t)}) \leq f(x_n^{(t)}) \\ x_n^{(t)}, & \text{otherwise} \end{cases} \tag{44}$$

where $f(\cdot)$ denotes the value of the objective function (38) for the corresponding individual.

Time Complexity of DE. The solution is obtained through Algorithm (2). Initially, the population is initialized with a size of $N \times M$, resulting in a time complexity of $O(N \cdot M)$. For an individual x_n, it undergoes mutation, crossover, and selection operations, with a time complexity of $O(M)$. With a total of N individuals, the complexity per iteration becomes $O(N \cdot M)$. Considering a total of G iterations, the overall time complexity amounts to $O(G \cdot N \cdot M)$. The complexity of the differential evolution algorithm is thus $O(max(N \cdot M, G \cdot N \cdot M))$. It is evident that $G \cdot N \cdot M > N \cdot M$. Hence, the time complexity of Algorithm (2) is $O(G \cdot N \cdot M)$.

5 EXPERIMENTS

In this section, we present simulation results for HierFAVG-DE by comparing the DE-FL approach with ACO-FL (ant colony optimization) algorithm, DRL-FL (deep reinforcement learning) algorithm and two static benchmarks (DE-CFL and DE-WFL) in the HierFAVG framework. ACO is a probabilistic algorithm for finding optimal paths in a graph, and it is a bionic algorithm inspired by the behaviour of ants foraging in nature. During foraging, ants always follow an optimal path from the nest to the food source. DRL learns by interacting with the environment in real time, learning from its failures and successes, maximising the cumulative reward value that the agent receives from the environment, and ultimately enabling the agent to learn the optimal strategy. DE-CFL is an offloading strategy by performing complete FL iterations with DE algorithm, while DE-WFL is an offloading strategy without performing any FL iterations by DE algorithm.

We first verify the observations from the convergence analysis and illustrate the advantages of the hierarchical FL architecture in local-edge-cloud offloading collaboration. Then, we investigate the effects of different parameters on the DE-FL method. In addition, we test the system stability under the DE-FL algorithm.

5.1 Environmental Setup

Numerical results are obtained through computer simulations with Python 3.8.13. We consider a topology of MDs between 100 and 1000, ESs between 10 and 100, one cloud server in the experimental environment. The other parameters considered in simulation are listed in Table 2.

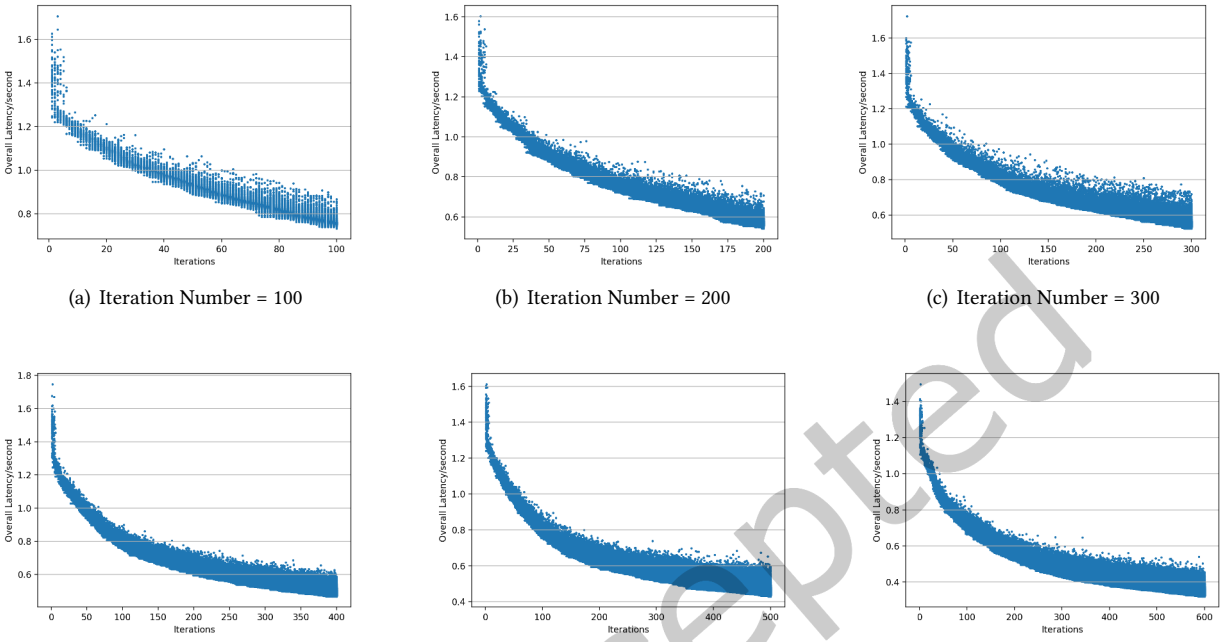
Table 2. Simulation Parameters

Parameter	Value
The mth MD's Flops $\tau_{d,m} \cdot f_{d,m}$	8 GFLOPs
The mth MD's Flops $\tau_{e,n} \cdot f_{e,n}$	80 GFLOPs
Allocated Bandwidth for the mth MD-ES Communication $B^E_{d,m}$	10 Mbit/s
Allocated Bandwidth for the mth MD-Cloud Communication $B^C_{d,m}$	100 Mbit/s
Task Size of the Task D_{x_m}	2.5 MB
Requested CPU Cycles of the Task Ω_{x_m}	2500 FLOPS
Task Result Size of the Task $D_{x_m,rx}$	0.5 MB
Propagation Speed in the Medium γ	10 km/s
Maximum Latency Allowed by the mth MD $\bar{T}_{x_m}$	2 s
The mth MD's Movement Speed v_m	[1.0, 1.5] m/s

5.2 Effectiveness Comparison Test

The algorithm's reliability can be verified through a convergence analysis in Figure 4. We evaluate the overall latency performance with DE-FL (proposed approach) under DE generations of 100-600. From the results in Figure 4 we witness that the overall latency will decrease mainly from 1.6s to 0.2s as the iterations increase. Meanwhile, we can find that the longer the iteration time is, the closer the offloading's overall latency tends to reach the ideal value. However, the final latency does not improve significantly with the increasing iteration number, and this is mainly because of the DE's fast convergence speed, Therefore, in the following experiments, we set the iteration number as 100, which can avoid irrelevant iteration traps.

We evaluate the task offloading's average latency, service time failures, sojourn time failures versus MD number and ES number under the local-edge-cloud collaboration environment. The experimental results are listed in Table 3.



(a) Iteration Number = 100 (b) Iteration Number = 200 (c) Iteration Number = 300

(d) Iteration Number = 400 (e) Iteration Number = 500 (f) Iteration Number = 600

Fig. 4. Impact of the service number on algorithms

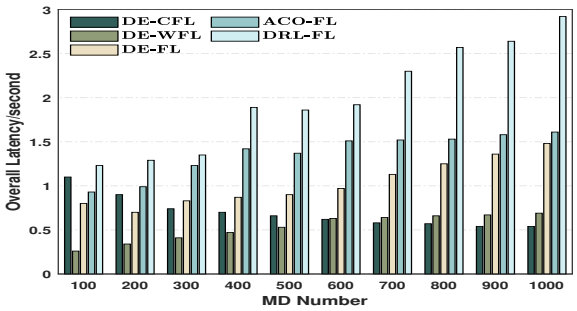
Table 3. Experimental Results

Performance		DE-CFL	DE-WFL	DE-FL	ACO-FL	DRL-FL
Average Latency	MD Number 100-1000	0.695	0.530	1.029	1.369	1.997
	ES Number 10-100	0.943	0.600	0.769	0.975	1.05
Service Time Failures	MD Number 100-1000	24.700%	1.790%	0	0	0
	ES Number 10-100	3.960%	0	0	0	0
Sojourn Time Failures	MD Number 100-1000	15.191%	12.327%	0	0	0
	ES Number 10-100	8.350%	6.363%	0	0	0

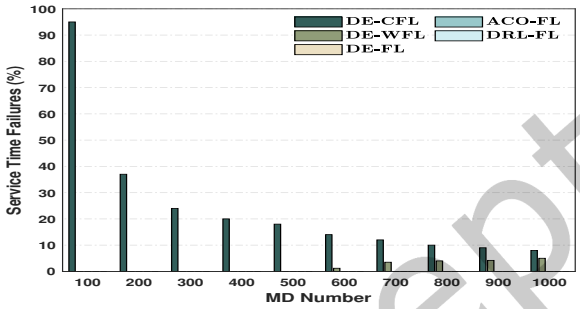
5.2.1 MD Number's Impact. In the following, we set the local-edge-cloud collaboration environment with 60 ESs and one Cloud, while MD number changes from 100 to 1000.

(1) Average Latency

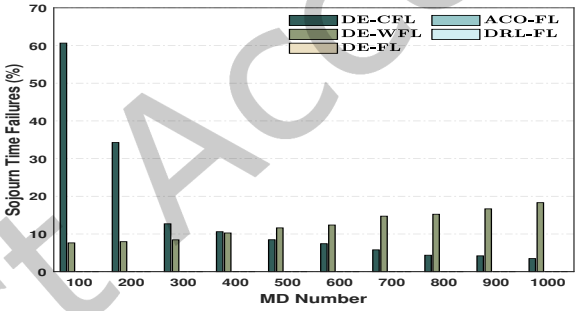
Here we discuss MD number's impact on normalized average latency as to different offloading approach: DE-CFL, DE-WFL, DE-FL, ACO-FL and DRL-FL. From the results in Figure 5(a) we witness that the DE-WFL approach performs better than the other approaches. When the MD number is less than 600, the overall latency



(a) Average Latency versus MD Number



(b) Service time Failures versus MD Number



(c) Sojourn Time Failures versus MD Number

Fig. 5. MD Number's Impact on Average Latency, Service Time Failures and Sojourn Time Failures

under the DE-WFL approach is always lower than DE-FL, DE-FL, ACO-FL and DRL-FL. After that, the DE-CFL approach begins to show its superiority, since FL process strength becomes increasingly apparent with an increasing number of MDs. On the other hand, the DE-FL approach with an adaptive FL process always performs better than ACO-FL, while DRL-FL takes the longest time. Their difference shows in searching modes, because compared to the DE algorithm, there are multiple search paths for the ACO algorithm, where each particle can actively seek the optimal solution. However, From an efficiency point of view, it is not necessary to involve every particle in searching, therefore, ACO-FL always needs more time to offload. Additionally, as the MD number increases in DRL-FL, the action space grows significantly, leading to the long latency.

(2) Service Time Failures

From the experimental results in Figure 5(b), we have two observations. First, comparing to the other four approaches, DE-CFL has a very high service failures, while the failure rate achieves over 80% when MD number is 100. After that, as the MD number increases, the service time failures under the DE-CFL approach will reduce rapidly. According to the aforementioned, in the FL process, the optimal FL iterations $\rho^{Best} = \frac{\mathcal{K}}{\overline{M}}$ whereas $\overline{M}$ is the participating MDs. ρ^{Best} can be easier achieved as the higher number of MDs involved, then, the optimal offloading decision can be obtained under the support from FL. Second, the service time failures under DE-WFL, DE-FL, ACO-FL and DRL-FL approaches tend to be 0, and this is because FL iterations need to consume a certain amount of time. DE-FL's, ACO-FL's, and DRL-FL's FL process is adaptive, so they can make an adjustment between task processing time and FL processing time, while DE-CFL and DE-WFL can't. Besides, when the MD number reaches 600 or more, service time failures appear in the DE-WFL without any FL's supports.

(3) Sojourn Time Failures

Figure 5(c) talks about the MD number's impact on sojourn time failures as to different approaches. Within MD number changing from 100 to 1000, the sojourn time failures under the DE-CFL approach decreases rapidly from 60% to 3%, while DE-FL, ACO-FL, and DRL-FL approaches generate no sojourn time failures. On the other hand, the DE-WFL approach's sojourn time failure rate increases from 7% to 18% as the MD number changes from 100-1000. The reasons are analysed as follows. First, if MDs spend a whole time on FL processing, which means there is no time left for task processing on MDs, thus the sojourn time constraint cannot be satisfied. Second, MDs have high mobility and interruptions to ESs' connection area will influence their task offloading process, which indirectly leads to sojourn time failures.

5.2.2 ES Number's Impact. In the following, we set the local-edge-cloud collaboration environment with 200 MDs and one Cloud, while ES number changes from 10 to 100.

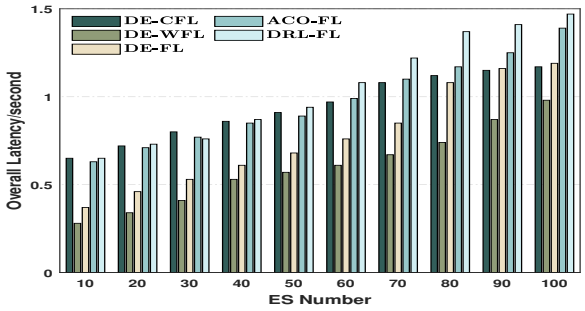
(1) Average Latency

Here we discuss ES number's impact on normalized average latency as to different offloading approaches: DE-CFL, DE-WFL, DE-FL, ACO-FL and DRL-FL. From the results in Figure 6(a) we witness that the DE-WFL approach performs the best among five approaches, which shows that the growth in ES number will weaken the benefits of FL processes in local-edge-cloud collaboration. This is participated because the partial model aggregation time will be stretched if more ESs join in. From the algorithm 1 we can see, the larger ES number N is, the more local parameters and more partial parameters need to be transmitted, and we need spend more time on waiting for the partial models to converge. On the other hand, ACO-FL approach performs bad because its search mode is not in a sufficient quality, while DRL-FL always needs longer latency due to the large state space. Besides, we cannot dismiss the DE-FL approach for the above reason, because in practical scenarios, so as to save on equipment costs, we always expect on deploying ESs as few as possible while can still solve the MDs' demands and achieve maximum benefits. Therefore, DE-FL is still a very promising approach.

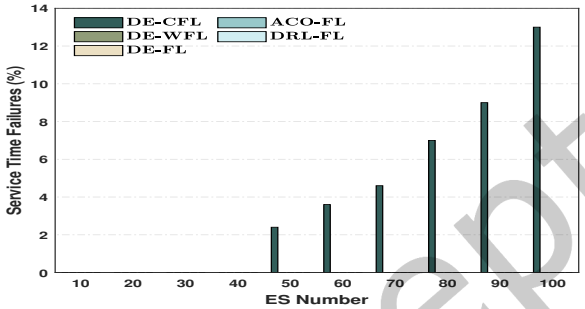
(2) Service Time Failures

From the experimental results in Figure 6(b), we have two observations. First, comparing to the other three approaches, DE-CFL has a very high service failures, while the failure rate increases from 0 to 13% when ES number transforms from 40 to 100. Meanwhile, as the ES number increases, the service time failures under the DE-CFL approach will increase as well. This finding is participated because performing complete FL iterations under large number of ESs takes so much time that the task processing time not enough. Since the service time constraint is fixed, the failures will significantly increase as the allocated task processing time decreases. Second, due to the adaptive characteristic of DE-FL's, ACO-FL's, and DRL-FL's FL processes, the task processing time under these three approaches can be well balanced, therefore, their service time failures tend to be 0. Besides, there are no service time failures under DE-WFL, because the FL process does not exist in this approach, and ES number's changing cannot have an impact on the DE-WFL approach.

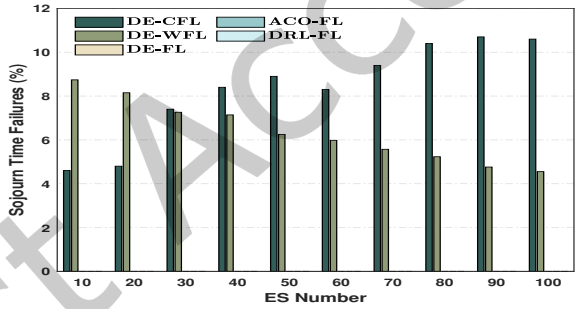
(3) Sojourn Time Failures



(a) Average Latency versus ES Number



(b) Service Time Failures versus ES Number



(c) Sojourn Time Failures versus ES Number

Fig. 6. ES Number's Impact on Average Latency, Service Time Failures and Sojourn Time Failures

Figure 6(c) talks about the ES number's impact on sojourn time failures as to different approaches. Within ES number changing from 10 to 100, the sojourn time failures under the DE-CFL approach increases from 4% to 12% and DE-WFL decreases from 9% to 4%, while DE-FL, ACO-FL, and DRL-FL approaches generate no sojourn time failures. The reasons are analysed as follows. First, as the aforementioned, the growth in ES number will weaken the benefits of FL processes on ESs when cooperating with task processing. From the algorithm 1 we can see, the larger ES number N is, the more local parameters and more partial parameters need to be transmitted, and we need spend more time on waiting for the partial models to converge. Hence, even the overall latency required will be affected, and then the task processing time allocated will be influenced to satisfy the sojourn time

constraint, which is prone to task failures. Second, as the MDs' positions constantly change, interruptions to ESs' connection area will have an impact on task offloading process, which indirectly leads to sojourn time failures.

5.2.3 DE-FL Iterations' Impact on Overall Latency. In the case of DE-FL, the latency performance can be improved by increasing the DE iteration number. In Figure 7, we compare the performance in terms of overall latency by considering a different number of DE iterations: 30, 50, 70, 90 and 100. It can be seen that as the number of iterations increases, the overall latency can be minimized.

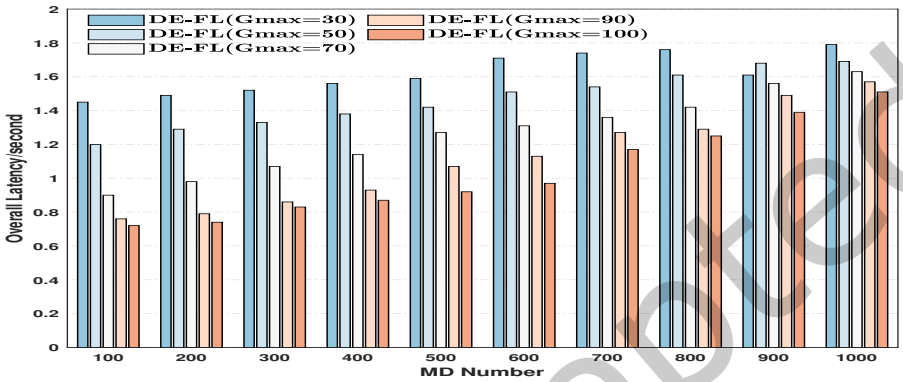


Fig. 7. DE-FL Performance Vs Iterations

6 CONCLUSION AND FUTURE WORK

In this paper, we develop a local-edge-cloud offloading strategy for latency and privacy sensitive tasks. Using a Hierarchical Federated Average method based on Differential Evolution algorithm (HierFAVG-DE), we conduct a three-layer hierarchical network architecture composed of Mobile Devices, Edge Servers and Cloud. With the help of Local, Edge, Cloud and Federated Models, we can get an optimal offloading decision in terms of overall latency. Moreover, simulations show that HierFAVG-DE significantly outperforms three other baseline approaches. Finally, we expect to build prediction models on top of this architecture for the next steps.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation of China (No. 62302200, No. 62125206, No. 62066019, No. U20A20173), the Natural Science Foundation of Jiangxi Province (No. 20232BAB212014), and the Technology Program of Jiangxi Education Department (No. GJJ2200831).

REFERENCES

[1] Lorenzo Campioni, Filippo Poltronieri, Cesare Stefanelli, Niranjan Suri, Mauro Tortonesi, and Konrad S. Wrona. 2023. Enabling civil-military collaboration for disaster relief operations in smart city environments. *Future Gener. Comput. Syst.* 139 (2023), 181–195. https://doi.org/10.1016/j.future.2022.09.020

[2] Baotong Chen, Jiafu Wan, Antonio Celesti, Di Li, Haider Abbas, and Qin Zhang. 2018. Edge computing in IoT-based manufacturing. *IEEE Communications Magazine* 56, 9 (2018), 103–109.

[3] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2015. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM transactions on networking* 24, 5 (2015), 2795–2808.

[4] Zong-Gan Chen, Zhi-Hui Zhan, Hua Wang, and Jun Zhang. 2019. Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems. *IEEE Trans. Evol. Comput.* 24, 4 (2019), 708–719.

[5] Shuiguang Deng, Zhengzhe Xiang, Javid Taheri, Mohammad Ali Khoshkholghi, Jianwei Yin, Albert Y Zomaya, and Schahram Dustdar. 2020. Optimal application deployment in resource constrained distributed edges. *IEEE Transactions on Mobile Computing* 20, 5 (2020), 1907–1923.

[6] Thinh Quang Dinh, Ben Liang, Tony QS Quek, and Hyundong Shin. 2020. Online resource procurement and allocation in a hybrid edge-cloud computing system. *IEEE transactions on wireless communications* 19, 3 (2020), 2137–2149.

[7] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[8] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[9] Hongzhi Guo and Jiajia Liu. 2018. Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks. *IEEE Transactions on Vehicular Technology* 67, 5 (2018), 4514–4526.

[10] Zicong Hong, Wuhui Chen, Huawei Huang, Song Guo, and Zibin Zheng. 2019. Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments. *IEEE Transactions on Parallel and Distributed Systems* 30, 12 (2019), 2759–2774.

[11] Jing Li. 2020. Resource optimization scheduling and allocation for hierarchical distributed cloud service system in smart city. *Future Generation Computer Systems* 107 (2020), 247–256.

[12] Jie Li, Yuxing Deng, Wei Sun, Weitao Li, Ruidong Li, Qiyue Li, and Zhi Liu. 2022. Resource Orchestration of Cloud-edge based Smart Grid Fault Detection. *ACM Transactions on Sensor Networks (TOSN)* (2022).

[13] Jing Li, Feng Xia, Wei Wang, Zhen Chen, Nana Yaw Asabere, and Huizhen Jiang. 2014. ACRec: a co-authorship based random walk model for academic collaboration recommendation. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel (Eds.). ACM, 1209–1214. https://doi.org/10.1145/2567948.2579034

[14] Wei Li, Xiang Meng, and Ying Huang. 2021. Fitness distance correlation and mixed search strategy for differential evolution. *Neurocomputing* 458 (2021), 514–525. https://doi.org/10.1016/j.neucom.2019.12.141

[15] Yangfan Li, Kenli Li, Wei Wei, Joey Tianyi Zhou, and Cen Chen. 2022. CoRec: An Efficient Internet Behavior based Recommendation Framework with Edge-cloud Collaboration on Deep Convolution Neural Networks. *ACM Transactions on Sensor Networks (TOSN)* (2022).

[16] Yan Liang, Xin Chen, Shengcheng Ma, and Libo Jiao. 2021. Delay-Sensitive Slicing Resources Scheduling Based on Multi-MEC Collaboration in IoV. In *Collaborative Computing: Networking, Applications and Worksharing - 17th EAI International Conference, CollaborateCom 2021, Virtual Event, October 16-18, 2021, Proceedings, Part II (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 407)*, Honghao Gao and Xinheng Wang (Eds.). Springer, 50–64. https://doi.org/10.1007/978-3-030-92638-0_4

[17] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B. Letaief. 2020. Client-Edge-Cloud Hierarchical Federated Learning. (2020), 1–6. https://doi.org/10.1109/ICC40277.2020.9148862

[18] Xinchen Lyu, Wei Ni, Hui Tian, Ren Ping Liu, Xin Wang, Georgios B Giannakis, and Arogyaswami Paulraj. 2017. Optimal schedule of mobile edge computing for internet of things using partial information. *IEEE Journal on Selected Areas in Communications* 35, 11 (2017), 2606–2615.

[19] Muhammet Oguz Ozcan, Fatih Odaci, and Ismail Ari. 2019. Remote debugging for containerized applications in edge computing environments. In *2019 IEEE international conference on edge computing (EDGE)*. IEEE, 30–32.

[20] Guanjin Qu, Naichuan Cui, Huaming Wu, Ruidong Li, and Yuemin Ding. 2022. ChainFL: A Simulation Platform for Joint Federated Learning and Blockchain in Edge/Cloud Computing Environments. *IEEE Trans. Ind. Informatics* 18, 5 (2022), 3572–3581. https://doi.org/10.1109/TII.2021.3117481

[21] Adalberto R Sampaio, Julia Rubin, Ivan Beschastnikh, and Nelson S Rosa. 2019. Improving microservice-based applications with runtime placement adaptation. *Journal of Internet Services and Applications* 10, 1 (2019), 1–30.

[22] You Shi, Changyan Yi, Bing Chen, Chenze Yang, Xiangping Zhai, and Jun Cai. 2022. Closed-Loop Control of Edge-Cloud Collaboration Enabled IIoT: An Online Optimization Approach. In *IEEE International Conference on Communications, ICC 2022, Seoul, Korea, May 16-20, 2022*. IEEE, 5682–5687. https://doi.org/10.1109/ICC45855.2022.9838906

[23] Swapnil Sadashiv Shinde, Arash Bozorgchenani, Daniele Tarchi, and Qiang Ni. 2022. On the Design of Federated Learning in Latency and Energy Constrained Computation Offloading Operations in Vehicular Edge Computing Systems. *IEEE Trans. Veh. Technol.* 71, 2 (2022), 2041–2057. https://doi.org/10.1109/TVT.2021.3135332

[24] Zhou Su, Yuntao Wang, Tom H. Luan, Ning Zhang, Feng Li, Tao Chen, and Hui Cao. 2022. Secure and Efficient Federated Learning for Smart Grid With Edge-Cloud Collaboration. *IEEE Trans. Ind. Informatics* 18, 2 (2022), 1333–1344. https://doi.org/10.1109/TII.2021.3095506

[25] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. 2022. PreGAN: Preemptive migration prediction network for proactive fault-tolerant edge computing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 670–679.

[26] Haoyu Wang, Jiaqi Gong, Yan Zhuang, Haiying Shen, and John Lach. 2017. Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 1213–1222.

[27] Pengfei Wang, Zijie Zheng, Boya Di, and Lingyang Song. 2019. HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing. *IEEE Transactions on Wireless Communications* 18, 10 (2019), 4942–4956.
[28] Yong Wang, Jie Zhang, Kevin Assogba, Yong Liu, Maozeng Xu, and Yinhai Wang. 2018. Collaboration and transportation resource sharing in multiple centers vehicle routing optimization with delivery and pickup. *Knowl. Based Syst.* 160 (2018), 296–310. https://doi.org/10.1016/j.knosys.2018.07.024
[29] Feng Wei, Sixuan Chen, and Weixia Zou. 2018. A greedy algorithm for task offloading in mobile edge computing system. *China Communications* 15, 11 (2018), 149–157.
[30] Fei Xu, Yue Xie, Yongyong Sun, Zengshi Qin, Gaojie Li, and Zhuoya Zhang. 2022. Two-stage computing offloading algorithm in cloud-edge collaborative scenarios based on game theory. *Comput. Electr. Eng.* 97 (2022), 107624. https://doi.org/10.1016/j.compeleceng.2021.107624
[31] Xiaolong Xu, Xihua Liu, Zhanyang Xu, Fei Dai, Xuyun Zhang, and Lianyong Qi. 2019. Trust-oriented IoT service placement for smart cities in edge computing. *IEEE Internet of Things Journal* 7, 5 (2019), 4084–4091.
[32] Min Xue, Huaming Wu, Guang Peng, and Katinka Wolter. 2022. DDPQN: An Efficient DNN Offloading Strategy in Local-Edge-Cloud Collaborative Environments. *IEEE Trans. Serv. Comput.* 15, 2 (2022), 640–655. https://doi.org/10.1109/TSC.2021.3116597
[33] Yanfen Xue, Guisheng Fan, Huiqun Yu, and Huaiying Sun. 2019. Energy-Aware Resource Scheduling with Fault-Tolerance in Edge Computing. In *IFIP International Conference on Network and Parallel Computing*. Springer, 327–332.
[34] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. 2020. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 393–399.
[35] Qingyang Zhang, Jie Cui, Hong Zhong, and Lu Liu. 2022. Toward Data Transmission Security based on Proxy Broadcast Re-encryption in Edge Collaboration. *ACM Transactions on Sensor Networks (TOSN)* (2022).
[36] Xiaokang Zhou, Wei Liang, Kevin I-Kai Wang, Zheng Yan, Laurence T. Yang, Wei Wei, Jianhua Ma, and Qun Jin. 2023. Decentralized P2P Federated Learning for Privacy-Preserving and Resilient Mobile Robotic Systems. *IEEE Wireless Communications* 30, 2 (2023), 82–89. https://doi.org/10.1109/MWC.004.2200381
[37] Xiaokang Zhou, Wei Liang, Ke Yan, Weimin Li, Kevin I-Kai Wang, Jianhua Ma, and Qun Jin. 2023. Edge-Enabled Two-Stage Scheduling Based on Deep Reinforcement Learning for Internet of Everything. *IEEE Internet of Things Journal* 10, 4 (2023), 3295–3304. https://doi.org/10.1109/JIOT.2022.3179231
[38] Xiaokang Zhou, Xiang Yang, Jianhua Ma, and Kevin I-Kai Wang. 2022. Energy-Efficient Smart Routing Based on Link Correlation Mining for Wireless Edge Computing in IoT. *IEEE Internet of Things Journal* 9, 16 (2022), 14988–14997. https://doi.org/10.1109/JIOT.2021.3077937
[39] Xiaokang Zhou, Xiaozhou Ye, Kevin I-Kai Wang, Wei Liang, Nirmal Kumar C. Nair, Shohei Shimizu, Zheng Yan, and Qun Jin. 2023. Hierarchical Federated Learning With Social Context Clustering-Based Participant Selection for Internet of Medical Things Applications. *IEEE Transactions on Computational Social Systems* 10, 4 (2023), 1742–1751. https://doi.org/10.1109/TCSS.2023.3259431
[40] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurence T. Yang, Shohei Shimizu, and Kevin I-Kai Wang. 2023. Digital Twin Enhanced Federated Reinforcement Learning With Lightweight Knowledge Distillation in Mobile Networks. *IEEE Journal on Selected Areas in Communications* 41, 10 (2023), 3191–3211. https://doi.org/10.1109/JSAC.2023.3310046