Ensuring Fairness in Edge Networks: A GNN-Based Media Workload Migration Scheme With Fairness Guarantee

Cheng Zhang, Jianwei Yin, and Shuiguang Deng, *Senior Member, IEEE*

***Abstract*— As the number of mobile and IoT devices grows, an edge network faces considerable difficulty in collaborative service provision. Computing and bandwidth resources in edge environments are restricted and unevenly distributed. Besides, video streaming requests from users may vary over time. These factors pose significant challenges for users to acquire a consistent Quality of Experience (QoE). Previous studies concentrate on a resource allocation problem with a single objective: maximizing total QoE for all users. Few studies have focused on a fairness issue for each user's QoE in an edge environment. This work designs a resource allocation algorithm by allocating computing and bandwidth resources fairly for all users. We leverage the relationships between entities in an edge network to model the graph embeddings of each entity using a graph neural network. Our proposed Path-embedding Learning (PEL) method learns from the network topology comprised of links with bandwidth resources and nodes with computing resources to generate feature representations of workload migration paths. Our proposed Fairness-ensure Backward Propagation (FBP) scheme can guarantee fairness for users. Our theoretical derivation reveals that the proposed algorithm achieves fairness with approximate Pareto optimality. Simulation results demonstrate that the algorithm can obtain better QoE than existing algorithms.**

***Index Terms*—Workload migration, graph neural network, fairness.**

I. Introduction

CURRENTLY, video streaming and video game services are among the most popular media on the Internet. According to Google statistics [1], nearly three out of five video views come from mobile devices. Moreover, the development of immersive videos, especially the augmented reality (AR) technique, is spawning a potential new generation of media applications that are both bandwidth-intensive and computing-intensive [2]. These enormous amounts of service traffic pose significant challenges to the network.

As a complement to current cloud computing, edge computing is closer to users or data sources. Edge computing arises to overcome the limitations of traditional cloud computing [3], [4], [5]. It provides users with low latency and gives them outstanding service experience. The media and video streaming framework assisted by edge computing is a promising architecture to facilitate content service providers to supply media and video streaming over Internet. The computing power of collaborative edge servers enables the migration of the media content efficiently to users and satisfies users' demands for different transcoding and transsizing [6], [7]. Raw media content processing at the edge servers can improve the response efficiency of user interaction-related services. Moreover, compressing and optimizing media content near users can also enhance data transmission efficiency and reduce costs from cloud centers. Therefore, **Media on edge** has attracted many companies to participate in developing media applications through edge computing, such as Netflix, Microsoft, and Tencent. This trend has resulted in the proliferation of edge architectures, for instance, Azure Media Services,[1] Media Services on AWS[2] and Broadcast Live.[3] Furthermore, the academic community has made noteworthy contributions to this field by introducing novel architectures, including a Dynamic Adaptive Streaming over HTTP (DASH) scheme, which aims to deliver high-quality media content through adaptive bitrate streaming in edge computing environments [8]. Despite that the edge computing techniques mentioned above primarily focus on service providers' deployment and performance optimization, there has been limited work that addresses personalized utility concerns from a user's perspective.

Limitations of resources in an edge network are yet a considerable challenge for service providers to guarantee QoE for all users. Video transcoding is a highly CPU-intensive process [6], and different requests for video transsizing leads to bandwidth resource demands dynamically [7]. Therefore, resource **rivalry** is prone among users in an edge network [6], [7], [9], [10], [11], [12], [13]. Video streaming service in an edge network urgently needs a fair strategy to mitigate impacts brought by the competition for computing and bandwidth resources among multiple users. Additionally, in resource provision problems,

Manuscript received 24 April 2023; revised 23 June 2023; accepted 15 July 2023. Date of publication 31 July 2023; date of current version 12 June 2024. This work was supported in part by the Key Research Project of Zhejiang Province under Grant 2022C01145 and in part by the National Science Foundation of China under Grant U20A20173 and Grant 62125206. Recommended for acceptance by H. Karatza. *(Corresponding author: Shuiguang Deng.)*

The authors are with the College of Computer Science, Zhejiang University, Hangzhou, Zhejiang 310027, China (e-mail: coolzc@zju.edu.cn; zjuyjw@zju.edu.cn; dengsg@zju.edu.cn).

Digital Object Identifier 10.1109/TSC.2023.3298695

[1][Online]. Available: https://azure.microsoft.com

[2][Online]. Available: https://aws.amazon.com/media-services/

[3][Online]. Available: https://www.alibabacloud.com/solutions/broadcast-live

there may be a marginal effect where the QoE received by a user does not increase linearly, but instead only marginally increases as more resources are allocated to them [14]. Thus, we need to allocate limited network resources to all users simultaneously such that each user can increase as much QoE as possible.

Existing media and video streaming studies focus on how edge computing can facilitate users' utilities by solving an optimization problem. Some scholars model all users' optimization problems towards a holistic view. Recent [6], [7], [15] only focused on the optimal resource allocation strategy for all users (e.g., devices and vehicles). However, the high level of diversity in each user's demand for video streaming raises the significant challenge that network resources, including bandwidth and computing resources, may not be allocated to each user **fairly**. Therefore, we propose a resource allocation scheme for a multi-access edge network by guaranteeing fairness among users. As a result, each user receives the maximized QoE, and no resource allocation change to any user could lead to an improvement in QoE unless decreasing the QoE for at least one user. We call this situation **Pareto optimality**.

To deeply explore the impact of topology on decision-making, we propose to use the graph neural network method to extract features from an edge network. Since there are heterogeneous nodes such as users, servers, and links in the edge network topology, we propose a heterogeneous graph neural network (GNN)-based approach to learn graph embeddings for various roles in an edge network. Furthermore, we have summarized two reasonable **meta-paradigms** of edge computing in the GNN algorithm. Finally, we offer a fairness-based backpropagation mechanism for neural network learning. We use the learned graph embedding feature information as the user decision model's input. Each user model is trained independently in **parallel** and generates its own decision for needed resources. Then, we aggregate the gradient of each user model into the final gradient for updating in each iteration.

This work aims to make the following novel contributions to the field of edge computing:

- We provide a **universal upper bound** for the multi-objective optimization problem of neural networks. With this upper bound, our algorithms can ensure an approximate **fairness** among all users.
- We propose two representative **GNN-based meta-paradigms** to reveal the edge computing paradigm's characteristics fully for the first time.
- We design a **distributed** multi-objective method that allows users to solve the allocation decision for video streaming in **parallel**, thus ensuring the method's overall efficiency. The results show that it outperforms them.

In addition, we conduct experiments to compare the proposed method with the existing ones.

Next, we begin by reviewing the related work in Section II. We introduce the system model and the GNN-based model design in Sections III and IV, respectively. In Section V, we present a theoretical analysis of the fairness guarantee. In addition, we conduct experiments to compare the proposed method with the existing ones in Section VI. The results show that it outperforms them. Finally, we conclude our work in Section VII.

II. Related Work

We survey related work in the following aspects: fairness guarantee, GNN-based feature learning, and distributed manner.

A. *Fairness Guarantee*

The network's utility is equivalent to the cumulative sum of the utilities of all users. Many studies widely use the scalarization technique to propose the multi-objective optimization problem of media and video streaming. Mahmoud et al. [16] design an approximation method to solve the non-convex network utility maximization (NUM) problem in the form of summarized users' utilities under the congestion condition. Yi-Hsuan et al. [15] use a game theory to maximize the summation of users' social welfare in the edge system. To maintain the global target of low variance for all vehicles, Fang et al. [6] aim at maximising the summation of each vehicle's utility function by performing optimal resource allocation for each vehicle. Li et al. [17] propose scheduling optimization problem by modeling the resource allocation of mobile application and the revenue of the cloud service supplier collaboratively for users. Recent fairness methods proposed by [18], [19] do not consider the dynamic of workload request for each user over a long period, and they regard the fairness problem as a convex problem which is unrealistic in most edge environment scenarios. These studies above cannot guarantee fairness for all users due to the lack of specified scalarization parameters according to the application scenario. Maximizing the overall QoE of all users may lead to a situation where some users get exceptionally high QoE, but others only receive low QoE. A resource allocation method may exist to allow all users to receive as many resources as possible fairly. Users with low QoE can receive additional resources by suitably lowering the resource utilization of high-QoE users. We present a Pareto-optimized multi-user resource allocation approach to meet this criterion so that all users can obtain a Pareto-optimal state.

B. *GNN-Based Feature Learning*

Many works use feature learning methods by GNN-based techniques for modeling and scheduling in edge networks. The work of [20] GNN to embed the state information of the edge-cloud system, including the clusters and the edge servers. The traffic routing embedding based on GNN in [21] learns the optimal path to deploy the service function chain in an edge environment. The network planning strategy in [22] employs a transformation method to abstract link information in the topology of a network and implements the GNN by link features. However, the proposed GNN-based embedding method omits the heterogeneous entities in the edge network, i.e., the algorithm in [20] builds graph embedding for servers, and algorithms in [21], [22] only focus on links in the network. The work in [23] provides a framework to implement Graph Neural Networks in networks and proposes a new concept called multi-stage message passing to deal with heterogeneous network entities. The authors in [24] perform graph embedding learning on edges and nodes as the basis for traffic routing selection.

But, studies of [23] and [24] omit the important heterogeneous graph characteristics when embedding where the neighboring nodes in high-order relations play a significant role in feature learning. Hence, we adopt augmented neighboring nodes to capture essential features of a heterogeneous edge network fully and propose meta-paradigms to learn features based on edge computing paradigms.

C. Distributed Manner

Some studies leverage the centralized algorithm to design resource allocation strategies for video streaming. The work in [13] proposes a livecast approach with a centralized framework to handle the transcoding and transmission at the first mile. Authors in [25] use a centralized constraints-aware controller to meet the user's dynamic demand. Moreover, studies for video streaming, e.g., the framework EVSO in [7], the algorithm DRL in [6], and NeuroPlan in [22], consider the centralized way to schedule the transcoding or transformation of video streaming in the edge network scenario. The centralized scheme is easy to implement, but its complexity is high compared to the distributed system. Distributed agents can release the burden of high computation when implementing resource allocation decisions compared with a centralized scheme. Hence, the distributed user decision model offers our designed algorithm a low complexity and flexibility approach. Each user decision model is responsible for its decision-making, efficiently guaranteeing fairness among users by a distributed aggregation manner.

In contrast to the above research, we propose a scheme to allocate resources that overcomes unfair allocations among users. We adopt a learning scheme that focuses on augmented neighboring nodes to tackle the challenges of heterogeneous edge networks. Furthermore, we leverage a distributed manner in our proposed algorithm to release the overall computation burden compared to the centralized scheduling method.

III. Preliminaries and Problem Definition

This section introduces a system model, including an edge server and user models. We then define the workload migration in detail with a fairness utility optimization as a multi-objective problem. In the field of edge computing, there are two distinct computing paradigms. The first one involves users performing their workloads by using an edge server located within the coverage range of a wireless access point [26], [27], [28], [29], [30]. Alternatively, users can take advantage of collaborative computing by migrating their workloads from one remote edge server to another within range of a user's wireless access point, effectively migrating workload [31], [32], [33], [34]. The workload in this paper refers to the required resources to perform at an edge server invoked by a user service request.

A. Edge Network Scenario

Video livecast frequently produce many computing-intensive and IO-intensive user requests [9], [13]. We use a simple example of an eSports game livecast [13] in the edge network to introduce the motivation for our proposed algorithm. Fig. 1 illustrates how the edge system handles the workload migration by considering an example of an edge server-assisted video livecast service. Suppose that edge server #1 does not deploy the specific livecast service to support user #1 due to the constrained resources at the edge server. Then, the system migrates the workload required for satisfying user #1 from edge server #1 to edge server #2 (server #2 has already deployed the specific video livecast service for user #1) via the migration path (the blue dotted arrows shown in Fig. 1), with the final video streaming produced by server #2 returning to user #1. User #1 and #2 will compete for the bandwidth resource in the coverage of edge server #1. Both users need as many resources as possible to receive higher QoE. After the proposed algorithms Fairness-ensure Backward Propagation (FBP) and Path-embedding Learning (PEL) allocate the bandwidth required by user #1 for the migration path and the computing resources from edge server #2, user #1 can enjoy the eSports game livecast service. User #2 can directly connect to edge server #1 for the eSports game livecast due to the available service for user #2 deployed at edge server #1. The bandwidth and computing resources are constrained. The main principle of our proposed algorithm is to make **Pareto Optimality** resource allocation decisions that are fairness guaranteed.

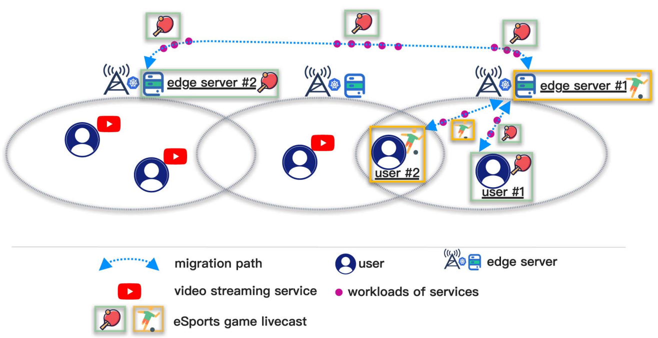

Fig. 1. Video streaming scenario in an edge network.

B. System Model

Edge Servers: We study this edge network by deployed edge servers $\mathcal{M} = \{1, 2, \ldots, M\}$, which provide services to nearby users. The edge network can accommodate various services $\mathcal{K}$, and the kth service provider will deploy its service at a set of candidate edge servers denoted by $\mathcal{M}_k, k \in \mathcal{K}$. We denote the computing resource at edge server j as C_j. Edge servers are connected in an edge network topology. Let $\hat{\mathcal{M}}_j$ be the set of edge servers connected to edge server $j, j \in \mathcal{M}$.

Edge Links: Denote the link between user i and server j as $e_{i,j}, i \in \mathcal{N}, j \in \mathcal{M}$. Let $\mathcal{E}$ be the set of user-to-server links. Since each edge server has a limited access bandwidth, the summation of wireless bandwidth used by users aggregated from the access point must satisfy the access bandwidth constraint $b_j, j \in \mathcal{M}$. We also denote the server-to-server link in an edge network as $l_{j,d}, j, d \in \mathcal{M}$, and $\mathcal{H}$ as the set of server-to-server links. We assume that link $l_{j,d}$ has a constrained bandwidth and limits maximum usage bandwidth to $b_{j,d}, d \in \hat{\mathcal{M}}_j$, where $\hat{\mathcal{M}}_j$ denote connected edge servers with edge server j.

Users: We use $\mathcal{N} = \{1, 2, \ldots, N\}$ to denote the set of users in an edge network. Since the coverage of wireless access points may overlap, user i in these overlapping areas can connect to any of the edge servers in this area. Denote these edge servers as $\bar{\mathcal{M}}_i$. Edge server j in $\bar{\mathcal{M}}_i$ can migrate the workload to another edge server $d, d \in \hat{\mathcal{M}}_j$ for user i in one hop. For edge server j, we denote users in its coverage as $\bar{\mathcal{N}}_j$. A group of users requests the same service k from $\mathcal{K}$ at time slot t. Let $\mathcal{N}_k$ be the users who invoke service request k.

Definition 1 (Migration Path): $P_i(e_{i,j})$ is a path along which user i migrates its workload to edge server $j, j \in \bar{\mathcal{N}}_j$. $P_i(e_{i,j}, l_{j,d})$ is a path along which the edge system migrates the workload of user i from edge server j to d by link $l_{j,d}$.

We denote the tuple of resources required to operate each workload of user i as $\boldsymbol{w}_i(t) = (w_{i,c}(t), w_{i,b}(t))$, where $i \in \mathcal{N}$. $w_{i,c}$ and $w_{i,b}$ stand for necessary computing resources and bandwidth resources, respectively. However, the resources allocated to each user may not equal the necessary resources required by the user. There exists competition for bandwidth resources and computing resources among all users. In addition, the workload generated by users is dynamic, and the computing and bandwidth resources allocated to each user may vary significantly from the user's demand. We use set notations $j \in \mathcal{M}_1, \mathcal{M}_1 = \mathcal{M}_i \cap \mathcal{M}_k, i \in \mathcal{N}_k, \forall k$ and $i \in \bar{\mathcal{N}}_j \cap \mathcal{N}_k, d \in \mathcal{M}_2, \mathcal{M}_2 = \mathcal{M}_k \cap \hat{\mathcal{M}}_j, \forall k$ to restrict user i only to migrate the workload via migration paths to edge servers that have already deployed the same service available for the user.

Definition 2 (Resources Allocation): $p^t_{i,c}(e_{i,j}), j \in \mathcal{M}_1, i \in \mathcal{N}_k, \forall k$ and $p^t_{i,c}(e_{i,j}, l_{j,d}), i \in \bar{\mathcal{N}}_j \cap \mathcal{N}_k, d \in \mathcal{M}_2, \forall k$ are the **computing resource** allocated to $P_i(e_{i,j})$ and $P_i(e_{i,j}, l_{j,d})$ for user i. $p^t_{i,b}(e_{i,j}), j \in \mathcal{M}_1, i \in \mathcal{N}_k, \forall k$ and $p^t_{i,b}(e_{i,j}, l_{j,d}), i \in \bar{\mathcal{N}}_j \cap \mathcal{N}_k, d \in \mathcal{M}_2, \forall k$ are the **bandwidth resource** allocated to paths $P_i(e_{i,j})$ and $P_i(e_{i,j}, l_{j,d})$ for user i.

We denote resource allocation for **migration paths** of user i as Ω_i, where $i \in \mathcal{N}$. We assume Ω_i in a fixed network should be finite and known in advance. By determining each element in $\Omega_i = \{p^t_{i,c}(e_{i,j}), \ldots, p^t_{i,b}(e_{i,j}), \ldots, p^t_{i,c}(e_{i,j}, l_{j,d}), \ldots, p^t_{i,b}(e_{i,j}, l_{j,d})\}$ for user i, we denote the bandwidth and computing resource allocated to user i as

$$x_{i,c}(t) = \sum_{j \in \mathcal{M}_1} p^t_{i,c}(e_{i,j}) + \sum_{j \in \mathcal{M}_1} \sum_{d \in \mathcal{M}_2} p^t_{i,c}(e_{i,j}, l_{j,d}), \quad (1)$$

$$x_{i,b}(t) = \sum_{j \in \mathcal{M}_1} p^t_{i,b}(j) + \sum_{j \in \mathcal{M}_1} \sum_{d \in \mathcal{M}_2} p^t_{i,b}(e_{i,j}, l_{j,d}), \quad (2)$$

where $p^t_{i,c}(e_{i,j}, l_{j,d}), p^t_{i,b}(e_{i,j}, l_{j,d}), p^t_{i,c}(e_{i,j}), p^t_{i,b}(e_{i,j})$ are non-negative.

C. Problem Definition

Let $x_{i,a}(t)$ be the total resource r allocated to user i at slot t, where $a \in \{c, b\}$ represents the computing or bandwidth resource. Each user has a utility function that measures the value of the respective computing and bandwidth resources obtained. Our proposed scheme encourages the system to provide a user with extra resources than the minimum requested, allowing the user to achieve higher QoE. In an edge computing scenario, as the resources acquired by a user increase, the user utility value should subsequently increase slowly. We adopt a log function $log(x_{i,r}(t) + 1)$ to represent this marginal effect, a typical utility measurement in resource-constrained edge networks [14]. However, if the user receives fewer resources than the minimum demanded, its QoE drops, and we introduce a penalty [35], [36] by a certain amount: $x_{i,r}(t) - w_{i,r}(t)$.

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$\mathcal{N}$	Set of users in the edge network
$\mathcal{M}$	Set of edge servers in the edge network
$\mathcal{E}$	Set of user-to-server links in the edge network
$\mathcal{H}$	Set of server-to-server links in the edge network
$\mathcal{N}_k$	Set of users invoked the same service request k
$\bar{\mathcal{N}}_j$	Set of users in the coverage of edge server j
$\mathcal{K}$	Set of services available in the edge network
$\mathcal{M}_k$	Set of edge servers with service k deployed
$\bar{\mathcal{M}}_i$	Set of edge servers that user i can connect
$\hat{\mathcal{M}}_j$	Set of edge servers connected to edge server j
$\boldsymbol{w}_i(t)$	Resources required by the workload from user i
x^t_i	Allocated resources for user i at time slot t
$P_i(e_{i,j})$	Paths for migrating the workload of user i
$P_i(e_{i,j}, l_{j,d})$	Paths for migrating the workload of user i
$p^t_{i,b}(e_{i,j})$	Allocated bandwidth resource for $P_i(e_{i,j})$
$p^t_{i,b}(e_{i,j}, l_{j,d})$	Allocated bandwidth resource for $P_i(e_{i,j}, l_{j,d})$
$p^t_{i,c}(e_{i,j})$	Allocated computing resource for $P_i(e_{i,j})$
$p^t_{i,c}(e_{i,j}, l_{j,d})$	Allocated computing resource for $P_i(e_{i,j}, l_{j,d})$

Let $x^t_i = \{x_{i,c}(t), x_{i,b}(t)\}$. Thus we define the utility function for user i as follows:

$$U^t_i(x^t_i) = \begin{cases} \sum_{a \in \{c,b\}} log(x_{i,a}(t) + 1), & x_{i,a}(t) \geq w_{i,a}(t), \\ \sum_{a \in \{c,b\}} (x_{i,a}(t) - w_{i,a}(t)), & x_{i,a}(t) < w_{i,a}(t). \end{cases} \quad (3)$$

Due to limited resources, edge server j only deploys a subset of services $\mathcal{K}$. We denote the services deployed at edge server j as $\mathcal{K}_j$. Any edge server $j \in \bar{\mathcal{M}}_i, i \in \mathcal{N}$ could migrate its workload requested by user i to connected edge server $d \in \hat{\mathcal{M}}_j$ only if edge server d deploys the service to take the workload. Then, we formulate a fairness optimization problem as follows:

$$\boldsymbol{\mathcal{P}}: \max_{\Omega_i, \forall i \in \mathcal{N}} \boldsymbol{U}, \quad (4)$$

$$\boldsymbol{s.t.}\ \boldsymbol{U} = \left\{ \cdots, \sum_{t \in \mathcal{T}} U^t_i(x^t_i), \cdots \right\}, \quad (5)$$

$$\sum_{i \in \bar{\mathcal{N}}_j} p^t_{i,b}(e_{i,j}) \leq b_j, \quad (6)$$

$$\sum_{i \in \bar{\mathcal{N}}_j} p^t_{i,b}(e_{i,j}, l_{j,d}) + \sum_{i \in \bar{\mathcal{N}}_d} p^t_{i,b}(e_{i,d}, l_{j,d}) \leq b_{j,d} \quad (7)$$

$$\sum_{i \in \bar{\mathcal{N}}_j} p^t_{i,c}(e_{i,j}) + \sum_{i \in \bar{\mathcal{N}}_d} \sum_{d \in \hat{\mathcal{M}}_j} p^t_{i,c}(e_{i,j}, l_{j,d}) \leq c_j, \quad (8)$$

$$x^t_i \geq 0, i \in \mathcal{N}, \forall j, d, \quad (9)$$

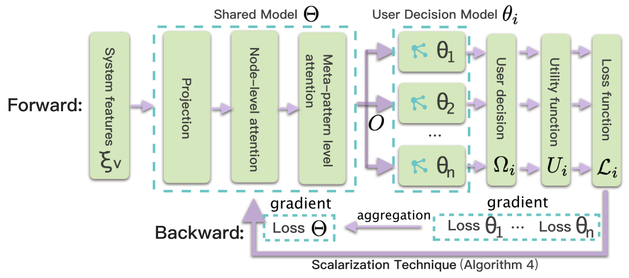

Fig. 2. The overall framework of the proposed Pareto optimization.

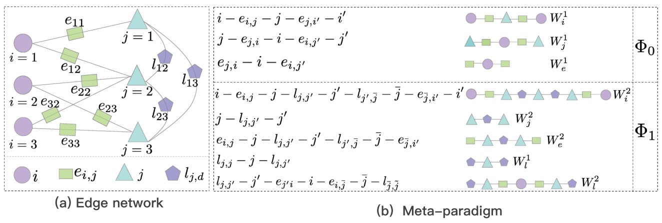
e_{11}
e_{12}
e_{22}
e_{32}
e_{23}
e_{33}
$i = 1$
$i = 2$
$i = 3$
$j = 1$
$j = 2$
$j = 3$
l_{12}
l_{13}
l_{23}
i
$e_{i,j}$
j
$l_{j,d}$
(a) Edge network
$i - e_{i,j} - j - e_{j,i'} - i'$ W_i^1
$j - e_{j,i} - i - e_{i,j'} - j'$ W_j^1
$e_{j,i} - i - e_{i,j'}$ W_e^1
Φ_0
$i - e_{i,j} - j - l_{j,j'} - j' - l_{j',\bar{j}} - \bar{j} - e_{\bar{j},i'} - i'$ W_i^2
$j - l_{j,j'} - j'$ W_j^2
$e_{i,j} - j - l_{j,j'} - j' - l_{j',\bar{j}} - \bar{j} - e_{\bar{j},i'}$ W_e^2
$l_{j,j} - j - l_{j,j'}$ W_l^1
$l_{j,j'} - j' - e_{j'i} - i - e_{i,\bar{j}} - \bar{j} - l_{\bar{j},\tilde{j}}$ W_l^2
Φ_1
(b) Meta–paradigm

Fig. 3. The graph topology of an edge network with meta-paradigms.

$$p_{i,c}^t(e_{i,j}) \geq 0, p_{i,b}^t(e_{i,j}) \geq 0, \tag{10}$$

$$p_{i,c}^t(e_{i,j}, l_{j,d}) \geq 0, p_{i,b}^t(e_{i,j}, l_{j,d}) \geq 0. \tag{11}$$

We utilize equation (5) to represent the user's utility function $U_i^t(x_i^t)$ as a vector $\mathbf{U}$ with components. Formula (6) specifies the bandwidth allocated to users in the coverage of edge server j should not exceed the limited access bandwidth. Formula (7) ensures the migrated workload between any two edge servers should not exceed the bandwidth limit $b_{j,d}$ for path $l_{j,d}$. Formula (8) indicates that the allocated computing resources for users within the service range of edge server j and other users whose load is migrated to j should be at most the upper limit c_j for edge server j's computing resources. Formulas (9)–(11) state that the allocated bandwidth and computing resources for users based on migration paths must be non-negative.

IV. GNN-Based Model Design

Problem Analysis: It is worth mentioning that the optimization problem is a multi-objective problem in which all users want to maximize their utilities. In other words, a fair edge network system does not arbitrarily penalize users' utilities. As shown in Fig. 2, to solve problem $\mathcal{P}$, we propose our strategy by using a GNN-based model and the optimization technique with the fairness guarantee. Each distributed user can decide on resource allocation by the proposed user decision model. We follow the rule of **Pareto Optimality** [19], [37], [38] for distributed multi-objective learning to design our proposed algorithm. Denote the loss function as $\mathcal{L}_i(\theta_i, \Theta; \xi)$ under stochastic requests ξ for user tasks. We need to increase the empirical utility of each user simultaneously. We use the scalarization technique to construct a minimized optimization problem by aggregating each user's loss function, i.e.,

$$\min_{i \in \mathcal{N}} \omega_i \mathcal{L}_i(\theta_i, \Theta), \tag{12}$$

where ω_i is a designed weight, and $\mathcal{L}_i(\theta_i, \Theta) = \mathbb{E}_\xi[\mathcal{L}_i(\theta_i, \Theta; \xi)]$ is the empirical loss function for user i. We aim to find feasible parameters $\theta^\star$ and $\Theta^\star$ of the proposed model such that all users' empirical loss functions meet the Pareto Optimality condition: $\mathcal{L}(\theta_i^\star, \Theta^\star) \leq \mathcal{L}(\theta_i, \Theta), \forall i \in \mathcal{N}$ and $\mathcal{L}(\theta_i^\star, \Theta^\star) < \mathcal{L}(\theta_i, \Theta), \exists i \in \mathcal{N}$. We refer to the utility of each user as being fairly assured when it satisfies the condition above.

We depict the utilization of graph embedding for network entities and each user decision model in Fig. 2. The input of system features consists of the computing resources and bandwidth resources utilized by the users, as well as the allocated computing resources from the servers and offered bandwidth resources from the user-to-server and server-to-server links in the edge network. These values can be obtained using monitoring tools provided by the cloud-native platform. The user decision model serves as a model for generating resource allocation decisions for individual users. Additionally, we can utilize either a capable edge server or a cloud server located outside the edge network to handle aggregating distributed gradients from user decision models.

A. Graph Embedding for Workload Migration Paths

Our GNN-based model consists of two components, a shared model and user decision models, illustrated in Fig. 2. The input of our GNN-based model is derived from the system information of an edge network, which is used to generate embedding vectors for entities within the network. These embedding vectors are capable of uncovering the fundamental features of the heterogeneous network topology. Subsequently, we employ these embedding vectors as inputs to each user decision model. An edge network consists of different nodes, edge servers, links, and users, forming a heterogeneous network topology. The connection across multiple nodes can reveal deep feature information hidden in a network topology. Fig. 3(a) depicts a graph topology of the edge computing network. We can identify a chain that ends with the same node type from a given starting node. This chain is considered a **high-order relation** [39] for the starting node, and nodes sharing the same **high-order relation** are viewed as **augmented neighboring nodes**. To denote a chain, we introduce the term **meta-paradigm**, created by inserting hyphens between the nodes in the chain.

Definition 3 (Meta-Paradigm): Given high-order relations in an edge network, $W_i^1 = i - e_{i,j} - j - e_{j,i'} - i'$, $W_i^2 = i - e_{i,j} - j - l_{j,j'} - j' - l_{j',\bar{j}} - e_{\bar{j},i'} - i'$, $W_j^1 = j - e_{j,i} - i - e_{i,j'} - j'$, $W_j^2 = j - l_{j,j'} - j'$, $W_e^1 = e_{j,i} - i - e_{i,j'}$, $W_e^2 = e_{j,i} - i - l_{i,j'} - j' - l_{j',\bar{j}} - e_{\bar{j},i'}$, $W_l^1 = l_{i,j} - j - l_{j,j'}$, $W_l^2 = l_{j,j'} - j' - e_{j'i} - i - e_{i,\bar{j}} - \bar{j} - l_{\bar{j},\tilde{j}}$, $i, i' \in \mathcal{N}, j, j', \bar{j}, \tilde{j} \in \mathcal{M}$, we say that $\{W_i^1, W_i^2\}$, $\{W_j^1, W_j^2\}$, $\{W_e^1, W_e^2\}$ and $\{W_l^1, W_l^2\}$ are meta-paradigms related to user i, user-to-server link $e_{j,i}$, edge server j and server-to-server link $l_{j,j'}$, respectively.

Inspired by the heterogeneous graph network [40], based on the essential characteristics of the computing paradigm (Section III), we use **meta-paradigms** in Definition 3 to classify different types of **high-order relations**.

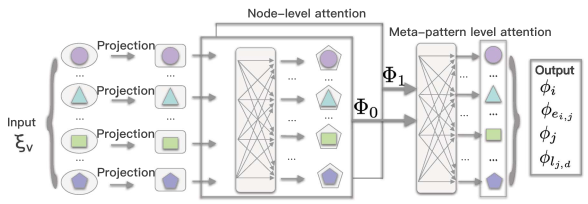

Fig. 4. Framework of the heterogeneous network embedding(shared model).

Meta-Paradigms are essential because:

- Users in $\mathcal{N}$ may implicitly characterize the joint resources usage of links and edge servers in an edge network.
- Other elements in $\{\mathcal{E} \cup \mathcal{M} \cup \mathcal{H}\}$ can implicitly describe the state of collaborative allocation of computing and bandwidth resources.

For instance, users $\#1$ and $\#2$ are **augmented neighboring nodes** because of **meta-paradigm** W_i^1, as shown in Fig. 3(a). Users $\#1$ and $\#2$ share the same server $\#2$'s computing resources and the access link's bandwidth resources of e_{12} and e_{22} owing to the topological connectivity. In other words, they would compete for the same bandwidth and computing resources. We can also find users $\#1$ and $\#3$ have the similar relation based on **meta-paradigm** W_i^1. Therefore, users $\#2$ and $\#3$ are **augmented neighboring nodes** of user $\#1$ under the premise of W_i^1.

As computing paradigms illustrated in Section III, we categorize **meta-paradigms** into two different groups, Φ_0 and Φ_1 and denote **meta-paradigm** as r, where $r \in \{\Phi_0, \Phi_1\}$. The primary difference between Φ_0 and Φ_1 is whether r contains a server-to-server link. We define $\Phi_0 = \{W_i^1, W_e^1, W_j^1\}$. Φ_0 illustrates the computing paradigm where users can complete their workload with the help of edge servers nearby. We define $\Phi_1 = \{W_i^2, W_e^2, W_j^2, W_l^1, W_l^2\}$. Φ_1 describes the computing paradigm where users must complete their workload with the help of an edge server within two hops.

We are now ready to present the method to fetch augmented neighboring nodes for each node in $\Gamma = \{\mathcal{N} \cup \mathcal{E} \cup \mathcal{M} \cup \mathcal{H}\}$, as shown in Algorithm 1. We define the set of augmented neighboring nodes for node $v \in \Gamma$ as $\Upsilon_r(v)$ according to meta-paradigm $r \in \{\Phi_0, \Phi_1\}$. First, we introduce an embedding algorithm for a heterogeneous network by a two-stage embedding corresponding to node level attention and meta-paradigm level attention. Second, we regard the previous round state of each node in Γ as dynamic features and the network topology as static features. Finally, our proposed GNN-based learning algorithm can leverage static and dynamic features to train a more accurate model. Fig. 4 presents the framework of this algorithm. This procedure is generalized in Algorithm 2.

1) Collecting Augmented Neighboring Nodes: Given the edge network topology in Fig. 3(a), we extract three bipartites $\mathcal{G}_{i,e_{i,j}}, \mathcal{G}_{e_{i,j},j}$, and $\mathcal{G}_{j,l_{j,d}}$, based on the network graph. These graphs represent the bipartite graph of users and user-to-links, the bipartite graph of user-to-links and edge servers, and the bipartite graph of edge servers and server-to-server links, respectively. We denote $A_{i,e_{i,j}}, A_{e_{i,j},j}$ and $A_{j,l_{j,d}}$ as the adjacency matrices of $\mathcal{G}_{i,e_{i,j}}, \mathcal{G}_{e_{i,j},j}$ and $\mathcal{G}_{j,l_{j,d}}$, respectively. From line #5 to #12 in Algorithm 1, we can calculate the adjacency matrix of nodes at both ends of a meta-paradigm under high-order relation. First, divide all elements in the meta-paradigm into a series of node pairs by the front and back order. Then, we continuously multiply the corresponding adjacency matrix according to each node pair to generate the final matrix. This new matrix is the adjacency matrix of augmented neighboring nodes. As a result, each meta-paradigm has an adjacency matrix that we can use to retrieve all of the augmented neighboring nodes associated with the meta-paradigm.

Algorithm 1: Augmented Nodes Selecting (NNS).

```
1  INPUT:
2  Adjacency matrices $A_{i,e_{i,j}}, A_{e_{i,j},i}$ of bipartites $\mathcal{G}_{i,e_{i,j}}$,
3  Adjacency matrices $A_{e_{i,j},j}, A_{j,e_{i,j}}$ of bipartites $\mathcal{G}_{e_{i,j},j}$,
4  Adjacency matrices $A_{j,l_{j,d}}, A_{l_{j,d},j}$ of bipartites $\mathcal{G}_{j,l_{j,d}}$,
5  for $r \in \Phi_0 \cup \Phi_1$ do
6      $k \leftarrow 0$
7      $T \leftarrow A_{r[k],r[k+1]}$
8      for $a, b \leftarrow r[k], r[k+1]$ do
9          $T = T \cdot A_{a,b}$
10         $k \leftarrow k+1$
11     end
12 end
13 Let $a, b$ be the first and last nodes in $r$
14 if $T[a,b] > 0$ then
15     $a$ and $b$ are augmented neighboring nodes
16 end
17 OUTPUT:
18 Augmented neighboring nodes $\Upsilon_r(v)$.
```

As a simple example, considering meta-paradigm $r = W_i^1$, we can get the adjacency matrix of user nodes in an augmented neighboring nodes form by multiplying $A_{i,e_{i,j}} \cdot A_{e_{i,j},j} \cdot A_{j,e_{i,j}} \cdot A_{e_{i,j},i} = A_{i,e_{i,j}} \cdot A_{e_{i,j},j} \cdot A^T_{e_{i,j},j} \cdot A^T_{i,e_{i,j}} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 2 \end{bmatrix}$. A non-zero element means that the two users represented by the row index and column one, respectively, are a pair of augmented neighboring nodes. Then, the result shows that all augmented neighboring nodes of user $\#1$ in Fig. 3(a) are users $\#2$ and $\#3$.

2) Node Level Attention: In line #3 in Algorithm 2, input ξ_v consists of the state of each entity in Γ of the topology. We briefly explain $\xi_v, v \in \Gamma$ as follows.

- For user i in $\mathcal{N}$, $\xi_i = [x_{i,c}(t), x_{i,b}(t)], i \in \mathcal{N}$ represents the bandwidth and computing resources allocated in the previous round.
- For user-to-server link $e_{i,j}$ in $\mathcal{E}$, $\xi_{e_{i,j}} = \sum_{i \in \mathcal{M}_j} p_i^b(j)$ represents the occupied bandwidth by all users on link $e_{i,j}$ in the previous round.
- For server-to-server link $l_{j,d}$ in $\mathcal{H}$, $\xi_{l_{j,d}} = \sum_{i \in \mathcal{N}} p_i^b(j, l_{j,d}, d)$ represents the occupied bandwidth by all users on migration link $l_{j,d}$ in the previous round.

- For edge server j in $\mathcal{M}$, $\xi_j = \sum_{i\in\mathcal{N}}(p_i^c(j) + p_i^c(d, l_{j,d}, j))$ represents the occupied computing resource by all users at edge server j in the previous round.

To make node feature have sufficient expressive power and be able to simultaneously calculate the characteristics of heterogeneous nodes in the same numerical space, we start by projecting different features of heterogeneous nodes to the same feature space. Line #8 in Algorithm 2 shows a projection process. By multiplying the transformation matrix W_r to ξ_v, we can get projected features $\bar{\xi}_v$ of node $v \in \{\mathcal{N}\cup\mathcal{E}\cup\mathcal{M}\cup\mathcal{H}\}$. We adopt a graph attention mechanism [41] to learn the weight of node n to v shown on line #9 by the normalization as:

$$\alpha_{v,n}^r = \frac{\exp(LeakyReLU(\vec{a}[W_r\xi_v \| W_r\xi_n]))}{\sum_{k\in\Upsilon_r(v)}\exp(LeakyReLU(\vec{a}[W_r\xi_v \| W_r\xi_k]))}, \tag{13}$$

where transform matrix W_r is a single-layer neural network with parameter $\vec{a}$, LeakyReLU is a nonlinear activation function [42], and $[W_r\xi_v \| W_r\xi_k]$ represents the concatenation operation in an attention mechanism [43].

After obtaining weight $\alpha_{v,n}^r$ of each pair nodes (v, n) in the same meta-paradigm r, we use a multi-head attention technique [44] to learn features further under the same meta-paradigm Φ. After line #11 repeats node level attention Y times, we can obtain that

$$\phi_v^r \leftarrow \|_{y=1}^{Y}\sigma\left(\sum_{n\in\Upsilon_r}\alpha_{v,n}^r\cdot\bar{\xi}_v\right), \tag{14}$$

where $\sigma(\cdot)$ is a nonlinear transformation and $r \in \{\Phi_0, \Phi_1\}$.

3) Meta-Paradigm Level Attention: Only using embeddings of nodes in each meta-paradigm is not enough to represent the features of a whole network. To obtain detailed and comprehensive feature information, we need to input the node level embedding $\alpha_{v,n}^r$ into the meta-paradigm level neural network for training to reveal the influence of different meta-paradigms on each node level embeddings [40].

First, in line #12, we compute the importance of each meta-paradigm by a transformation matrix W_r', $r \in \Phi$. We denote the average importance of node level embeddings as:

$$\varpi(\mathcal{V}, r) = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\boldsymbol{q}^T\cdot\tanh(W_r'\cdot\phi_v^r + \boldsymbol{b}), \tag{15}$$

where $\mathcal{V} \in \{\mathcal{N}, \mathcal{E}, \mathcal{M}, \mathcal{L}\}$, $\boldsymbol{q}$ is the meta-paradigm level attention vector and $\boldsymbol{b}$ is the bias vector.

Second, we normalize the importance $\varpi(\mathcal{V}, r)$(from (15)) of each meta-paradigm level by function softmax, i.e.,

$$\varpi(\mathcal{V}, r) = \frac{\exp(\varpi(\mathcal{V}, r))}{\exp(\varpi(\mathcal{V}, r)) + \exp(\varpi(\mathcal{V}, r'))}, \tag{16}$$

$$\varpi(\mathcal{V}, r') = \frac{\exp(\varpi(\mathcal{V}, r'))}{\exp(\varpi(\mathcal{V}, r)) + \exp(\varpi(\mathcal{V}, r'))} \tag{17}$$

where $r \in \Phi_0$ and $r' \in \Phi_1$ for nodes $\mathcal{V} \in \{\mathcal{N}\cup\mathcal{E}\cup\mathcal{M}\}$, and $r, r' \in \Phi_1$ for nodes $\mathcal{V} = \mathcal{H}$. We then fuse the meta-paradigm embeddings by the normalized weight $\varpi(\mathcal{V}, r)$ of each node on line #13. We denote ϕ_v as the final embedding of node $v \in \{\mathcal{N}\cup\mathcal{E}\cup\mathcal{M}\cup\mathcal{H}\}$:

$$\phi_v = \varpi(\mathcal{V}, r)\cdot\phi_v^r + \varpi(\mathcal{V}, r')\cdot\phi_v^{r'}. \tag{18}$$

Finally, we can concatenate the node embedding ϕ_v to form a migration path. As shown in a simple example, user #1 in Fig. 3 can migrate workload to edge server #1 and edge server #2 by migration paths $P_1(e_{1,1})$ and $P_1(e_{1,1}, l_{1,2})$. Then, by concatenating the outputs shown in Fig. 4, we obtain the path embeddings $[\phi_i, \phi_{e_{1,1}}, \phi_j]$ and $[\phi_i, \phi_{e_{1,1}}, \phi_j, \phi_{l_{1,2}}, \phi_{j'}]$, where $i = 1, j = 1, j' = 2$.

4) User Decision Model: Each user acts as an independent agent, deploying a decision model, as shown in Fig. 5. Let the embedding of all migration paths O be the model's input, and then, we have the output of the resource allocation Ω_i, $i \in \mathcal{N}$ for user i's migration path. In this paper, we design a simple learning network for each user decision model. Improving the network model of user decision-making is interesting but not the focus of this paper. However, it can be a future research problem.

During forward propagation, users receive embeddings of migration paths from the shared model as the input of its decision model. Then, all users' weighted gradients aggregate to a new gradient for the backward propagation of the shared model for learning. The output Ω_i consists of the set of decisions $\{p_{i,c}^t(e_{i,j}), \ldots, p_{i,b}^t(e_{i,j}), \ldots, p_{i,c}^t(e_{i,j}, l_{j,d}), \ldots, p_{i,b}^t(e_{i,j}, l_{j,d})\}$ for user i. Suppose that Θ is the parameter of the shared model and θ_i is the parameter of a user decision model for user i.

Algorithm 2: Path-embedding Learning (PEL).

```
1  INPUT:
2  Nodes of the topology: $\Gamma \leftarrow \{\mathcal{N}\cup\mathcal{E}\cup\mathcal{M}\cup\mathcal{H}\}$
3  The node feature set: $\{\xi_v | v \in \Gamma\}$,
4  The meta-paradigm set: $\Phi = \{\Phi_0, \Phi_1\}$.
5  The number of multi-head attention: $Y$
6  Augmented neighboring nodes: $\Upsilon_r \triangleleft$ Algorithm 1.
7  for $v \in \Gamma$ do
8      $\bar{\xi}_v \leftarrow W_r\xi_v$,
9      Calculate weight $\alpha_{v,n}^r, r \in \Phi$ for node $v$ by (13).
10 end
11 Concatenate node attentions $\phi_v^r$ by (14).
12 Compute normalized meta-paradigm weights $\varpi(\mathcal{V}, r)$
     and $\varpi(\mathcal{V}, r')$ by (16) and (17).
13 Fuse embeddings $\phi_v$ for node $v$ by (18).
14 Learning by gradient descent $\triangleleft$ Algorithm 3.
15 OUTPUT:
16 Concatenate the final embedding of node $\phi_v$ into
     output $O$, representing embeddings of all migration
     paths
```

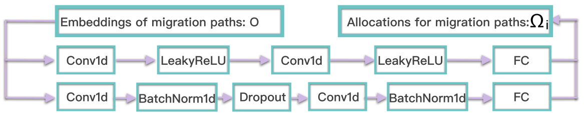
Embeddings of migration paths: O
Allocations for migration paths: Ω_i
Conv1d
LeakyReLU
Conv1d
LeakyReLU
FC
Conv1d
BatchNorm1d
Dropout
Conv1d
BatchNorm1d
FC

Fig. 5. User decision model.

We next introduce how to optimize weighted gradients **fairly** during backward propagation.

We define the loss function for each user as $\mathcal{L}_i(\theta_i, \Theta) =$

$$\begin{cases} \sum_{t'=0}^{t-1} U_i^{t'}(x_i^{t'}) + \gamma/(1+\gamma \cdot U_i^t(x_i^t)), & U_i^t(x_i^t) > 0, \\ \sum_{t'=0}^{t-1} U_i^{t'}(x_i^{t'}) + Exp(-U_i^t(x_i^t)) + \gamma - 1, & U_i^t(x_i^t) < 0, \end{cases} \tag{19}$$

where $U_i^t(x_i^t)$ can be calculated during each iteration in the model training, and $\gamma > 0$ is a pre-determined parameter in our training which guarantees the continuity of the piecewise function $\mathcal{L}_i(\theta_i, \Theta)$.

When designing the loss function, we follow that utility $U_i^t(x_i^t)$ in the iterative process should be positive and as large as possible. Therefore,

1) $U_i^t(x_i^t) \geq 0$: As $U_i^t(x_i^t)$ increases, the loss function $\mathcal{L}_i(\theta_i, \Theta)$ should approach 0. Thus, we set the loss function as $\gamma/(1+\gamma \cdot U_i^t(x_i^t))$.
2) $U_i^t(x_i^t) < 0$: As $U_i^t(x_i^t)$ becomes smaller, the loss function $\mathcal{L}_i(\theta_i, \Theta)$ should tend to be positive infinity. Thus, we set the loss function as $Exp(-U_i^t(x_i^t)) + \gamma - 1$.

B. Fairness Guarantee for All Users

Given the formulation of problem $\mathcal{P}$, we design FBP to process the GNN-based Backward propagation as illustrated in Algorithm 3. Since all objectives conflict with each other, for instance, shown in Fig. 1, increasing the bandwidth resource of user #1 lowers the remaining bandwidth resource shared by users in the signal coverage of edge server #1. Designing an algorithm to optimize all objectives simultaneously is non-trivial. During the training process, it is hard to obtain the optimal Pareto solution for each user in advance. Therefore, we design the training for path embedding as an unsupervised learning process.

Multi-Objective Gradient Descent: Fig. 2 shows that the edge system updates the parameters of the shared model by a scalarization technique. To design how the combination of all loss functions from user decision models is a critical problem. For simplicity, we use $\mathcal{L}_i$ to represent loss function $\mathcal{L}_i(\theta_i, \Theta)$ for user i. For multi-objective optimization algorithms, we can manually set the step size of a gradient descent method. Let ω_i be the weight of the gradient descent of user loss function $\nabla_\Theta \mathcal{L}_i$ w.r.t the parameter of the shared model. Determining weight $\omega_i, i \in \mathcal{N}$ for each gradient descent is a crucial issue for fairness guarantee.

Using a learning rate η, we merge all gradient descent of user loss functions and denote $\sum_{i\in\mathcal{N}} \omega_i \cdot \nabla_\Theta \mathcal{L}_i$ as the gradient descent of the loss function of the shared model. The Pareto efficient optimization method MGDA [37] proves that

$$\sum_{i\in\mathcal{N}} \omega_i \cdot \nabla_\Theta \mathcal{L}_i = 0. \tag{20}$$

holds, together with following constraints:

$$\sum_{i\in\mathcal{N}} \omega_i = 1, \tag{21}$$

$$\forall \omega_i \geq 0, i \in \mathcal{N} \tag{22}$$

Algorithm 3: Fairness-Ensure Backward Propagation (FBP).

```
1  INPUT:
2  Loss functions of user $i$: $\mathcal{L}_i, \forall i \in \mathcal{N}$,
3  Initialize the weights $\omega_i, \forall i \in \mathcal{N}$ uniformly ,
4  for $t \leftarrow 0$ to $\mathcal{T}$ do
5      for $i \leftarrow 0$ to $\mathcal{N}$ do
6          $\theta_i = \theta_i - \eta \cdot \nabla_{\theta_i} \mathcal{L}_i$ by (19)
7      end
8      Compute $\boldsymbol{\omega}^\star = \{\omega_1, ..., \omega_N\}$   ◁ Algorithm 4
9      $\Theta = \Theta - \eta \cdot \sum_{i\in\mathcal{N}} \omega_i \cdot \nabla_\Theta \mathcal{L}_i$
10 end
11 OUTPUT:
12 Updated parameter $\Theta$ for the path embedding model
     and $\theta_i$ for user $i$'s decision model.
```

can guarantee fairness for each user. Then, we follow the rule of minimizing the norm of a convex hull [37] constructed by (21) and (22). By introducing the quadratic form of (20), we have that

$$\left\| \sum_{i\in\mathcal{N}} \omega_i \cdot \nabla_\Theta \mathcal{L}_i \right\|_2^2 = \left(\sum_{i\in\mathcal{N}} \omega_i \cdot \nabla_\Theta \mathcal{L}_i \right)^T \left(\sum_{i\in\mathcal{N}} \omega_i \cdot \nabla_\Theta \mathcal{L}_i \right) \tag{23}$$

We denote the outputs of the shared model as O. According to the Encoder-Decoder Architecture in [38], we can obtain

$$\begin{aligned} \left\| \sum_{i\in\mathcal{N}} \omega_i \nabla_\Theta \mathcal{L}_i \right\|_2^2 &= \left\| \sum_{i\in\mathcal{N}} \omega_i \nabla_O \mathcal{L}_i \cdot \frac{\partial O}{\partial \Theta} \right\|_2^2 \\ &\leq \left\| \sum_{i\in\mathcal{N}} \omega_i \nabla_O \mathcal{L}_i \right\|_2^2 \cdot \left\| \frac{\partial O}{\partial \Theta} \right\|_2^2 \end{aligned} \tag{24}$$

The value $\|\frac{\partial O}{\partial \Theta}\|_2^2$ is not a funciton of $\boldsymbol{\omega}$. Thus we remove this part. By the upper bound $\|\sum_{i\in\mathcal{N}} \omega_i \nabla_O L_i\|_2^2$, we can relax (20) in the form of quadratic programming as

$$\begin{aligned} \boldsymbol{\mathcal{P}_1}: \quad & \min_{\boldsymbol{\omega}} \boldsymbol{\omega}^T \boldsymbol{B}\boldsymbol{B}^T \boldsymbol{\omega} \\ \boldsymbol{s.t.} \quad & (21), (22), \end{aligned}$$

where

$$\boldsymbol{\omega} = [\omega_1, \ldots, \omega_N]^T, \boldsymbol{B} = [\nabla_O \mathcal{L}_1^T, \ldots, \nabla_O \mathcal{L}_N^T]^T \tag{25}$$

V. Theoretical Analysis of Fairness Guarantee

Solving (20) is equivalent to solving the quadratic programming if there exists $\omega_i, \forall i \in \mathcal{N}$ such that the target function (20) equals 0. It has been proved that any solution of the minimum-norm $\boldsymbol{\mathcal{P}_1}$ satisfying (20)–(22) is also a Pareto stationary [37], which are necessary for Pareto optimality. Besides, any solution from $\boldsymbol{\mathcal{P}_1}$ satisfying (21)–(22) is acceptable to all users as a gradient descent direction for updating $\theta_i, \forall i \in \mathcal{N}$ increases the utility of all users. As a result of this conclusion, any optimized solution of problem $\boldsymbol{\mathcal{P}_1}$ is feasible to the edge network. Our target is to solve $\mathcal{P}_1$ and use the solution $\boldsymbol{\omega}^\star$ to update parameter Θ of the shared model.

The Stochastic Gradient (SG) method may lead to local optima in neural network training [45]. It has a higher chance of getting stuck in local optima when the multiple loss functions are used, as it increases the likelihood of at least one loss function gradient being zero. Based on the value of loss function $\nabla_O \mathcal{L}_i$, we summarize that the solution has two conditions, $\mathcal{S}_1$ and $\mathcal{S}_2$, given below.

- $\mathcal{S}_1$: $\nabla_O \mathcal{L}_{\tilde{i}} \neq 0, \nabla_O \mathcal{L}_{\hat{i}} = 0$ where $\tilde{i} \in \tilde{\mathcal{N}}, \hat{i} \in \hat{\mathcal{N}}$, $\tilde{\mathcal{N}} \cup \hat{\mathcal{N}} = \mathcal{N}, \tilde{\mathcal{N}} \cap \hat{\mathcal{N}} = \emptyset$, and $\hat{\mathcal{N}}$ is non-empty.
- $\mathcal{S}_2$: $\nabla_O \mathcal{L}_i \neq 0$, where $\forall i \in \mathcal{N}$

We analyze these two situations in detail as follows.

1) $\{\nabla_O \mathcal{L}_i\}$ Satisfies $\mathcal{S}_1$: We can easily get the optimal solution $\boldsymbol{\omega}^\star$ by using the following result.

Lemma 1: Suppose $\tilde{\boldsymbol{\omega}}$ and $\hat{\boldsymbol{\omega}}$ consist of elements $\omega_{\tilde{i}}, \tilde{i} \in \tilde{\mathcal{N}}$ and $\omega_{\hat{i}}, \hat{i} \in \hat{\mathcal{N}}$ respectively. We initialize $\omega_{\hat{i}}$ to 0 and apply Algorithm 4 to compute the optimal value of $\tilde{\boldsymbol{\omega}}$ (i.e., $\tilde{\boldsymbol{\omega}}^\star$), which satisfy $\mathcal{S}_2$. We can obtain $\boldsymbol{\omega}^\star$ by retrieving the corresponding component from vectors $\tilde{\boldsymbol{\omega}}^\star$ and $\hat{\boldsymbol{\omega}}$ according to user index $i \in \mathcal{N}$.

Proof: If $\hat{\mathcal{N}} \neq \emptyset$, there exists at least one user index $\hat{i}$ that $\nabla_O \mathcal{L}_{\hat{i}} = 0, \hat{i} \in \hat{\mathcal{N}}$. Let $\tilde{\boldsymbol{B}}$ be a vector with the $\hat{i}$th component of $\boldsymbol{B}$ removed. Repeatedly remove the $\hat{i}$th element in $\boldsymbol{B}$ if $\nabla_O \mathcal{L}_{\hat{i}} = 0$ until all elements in $\hat{\mathcal{N}}$ have been traversed. Finally, we can obtain an updated vector $\tilde{\boldsymbol{B}}$ from $\boldsymbol{B}$. Due to $\omega_{\hat{i}} \nabla_O \mathcal{L}_{\hat{i}} = 0, \forall \hat{i} \in \hat{\mathcal{N}}$, $\tilde{\boldsymbol{\omega}}^T \tilde{\boldsymbol{B}} = \sum_{\tilde{i} \in \tilde{\mathcal{N}}} \omega_{\tilde{i}} \cdot \nabla_O \mathcal{L}_{\tilde{i}}^T$ is equivalent to $\boldsymbol{\omega}^T \boldsymbol{B} = \sum_{i \in \mathcal{N}} \omega_i \cdot \nabla_O \mathcal{L}_i^T$. Thus, we conclude that $\boldsymbol{\omega}^{\star T} \boldsymbol{B}\boldsymbol{B}^T \boldsymbol{\omega}^\star = \tilde{\boldsymbol{\omega}}^T \tilde{\boldsymbol{B}} \tilde{\boldsymbol{B}}^T \tilde{\boldsymbol{\omega}}$. By finding $\tilde{\boldsymbol{\omega}}$ as an optimal solution of $\mathcal{P}_1$ with $\boldsymbol{B}$ replaced by $\tilde{\boldsymbol{B}}$, we can obtain $\boldsymbol{\omega}^\star$ from $\tilde{\boldsymbol{\omega}}$ and $\hat{\boldsymbol{\omega}}$ according to user index $i \in \mathcal{N}$. □

2) $\{\nabla_O \mathcal{L}_i\}$ Satisfies $\mathcal{S}_2$ Condition $\mathcal{S}_2$ leads to an optimal solution $\boldsymbol{\omega}^\star$ by Algorithm 3.

According to the necessary condition for a positive definite matrix, if $\boldsymbol{B}$ is not full-rank, then matrix $\boldsymbol{B}\boldsymbol{B}^T$ must not be positive definite [46]. However, the assumption that $\boldsymbol{B}$ is full-rank may not always hold in the actual scenario. Unlike Theorem in [38], we consider the assumption that $\boldsymbol{B}$ being full-rank does not necessarily hold and propose an improved bound of that in [38]. In this paper, we remedy the situation by setting $\boldsymbol{B}\boldsymbol{B}^T$ to a positive definite matrix when $\boldsymbol{B}$ is not full-rank. Furthermore, we derive the improved upper bound $\boldsymbol{\omega}^T \boldsymbol{H} \boldsymbol{\omega}$ to $\mathcal{P}_1$ after relaxation by designing a perturbation matrix, D. And we prove that the gap between the upper bound and optimal value of $\mathcal{P}_1$ is $\boldsymbol{\omega} D \boldsymbol{\omega}$ for all non-zero $\boldsymbol{w}$. Hence, when $\boldsymbol{B}\boldsymbol{B}^T$ is not positive definite, we employ a method similar to matrix perturbation [47] to replace it with a positive definite matrix by introducing a small perturbation to its eigenvalues. Denote the new positive definite matrix $\boldsymbol{H}$ as $\boldsymbol{B}\boldsymbol{B}^T + D$ where D is our designed perturbation matrix. The principle to design perturbation matrix D is to shift all the diagonal entries by the same amount. The diagonal entries of $\boldsymbol{B}\boldsymbol{B}^T$ are positive due to $\|\nabla_O \mathcal{L}_i\|_2^2 > 0$ (the solution satisfies $\mathcal{S}_2$), where $\forall i \in \mathcal{N}$.

Due to the positive definite matrix $\boldsymbol{B}\boldsymbol{B}^T + D$, we convert problem $\mathcal{P}_1$ to a Convex Quadratic Programming, which can be efficiently solved by well-known numerical optimization

Algorithm 4: Perturbed Scalarized Weights (PSW).

```
1  INPUT :$\boldsymbol{B}$ from (25)
2  if $\hat{\mathcal{N}} \neq \emptyset$ then
3      $\boldsymbol{B} = \tilde{\boldsymbol{B}}$ ◁ Lemma. 1
4  end
5  if $\boldsymbol{B}\boldsymbol{B}^T$ is not positive definite then
6      Compute Perturbation matrix $D$ by (26):
7      $\boldsymbol{B}\boldsymbol{B}^T \leftarrow \boldsymbol{B}\boldsymbol{B}^T + D$
8  end
9  Solve the problem $\mathcal{P}_1$ using Interior-Point Method to
     find $\boldsymbol{\omega}^\star$
10 if $\hat{\mathcal{N}} \neq \emptyset$ then
11     Retrieve $\boldsymbol{\omega}^\star$ by Lemma. 1
12 end
13 OUTPUT : $\boldsymbol{\omega}^\star$
```

methods. In Algorithm. 4, we use Interior-Point Method [48] by CVXOPT [49] to directly solve the optimization problem and $\boldsymbol{\omega}^\star$ is the unique global solution.

We provide a detailed illustration of how to design a perturbation matrix to relax $\mathcal{P}_1$ when the positive definiteness of $\boldsymbol{B}\boldsymbol{B}^T$ does not hold. To achieve this, we design a perturbation matrix D by using the following result.

Theorem 1 (Relaxation of $\boldsymbol{B}\boldsymbol{B}^T$): Let matrix Z be a diagonal matrix $\mathbf{diag}(\boldsymbol{B}\boldsymbol{B}^T)$, where $\mathbf{diag}(\cdot)$ outputs the diagonal matrix of the input matrix. Suppose that κ is the smallest eigenvalue of matrix

$$Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2}.$$

If we set perturbation matrix D as

$$(\epsilon - \kappa) \cdot Z, \tag{26}$$

where ϵ is a small positive real number. Then $\boldsymbol{H}$ is positive definite, where $\boldsymbol{H} = \boldsymbol{B}\boldsymbol{B}^T + D$.

Proof: Note that, each diagonal entry in Z is positive $\|\nabla_O \mathcal{L}_i\|_2^2 > 0$ ommitting 0 in Subspace $\mathcal{S}_2$. Hence, we can obtain that the determinant $|Z| = \prod_{i=1}^N \|\nabla_O \mathcal{L}_i\|_2^2 > 0$, and Z is positive definite. Note that $Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2}$ is a symmetric matrix. Its eigenvalues should satisfy

$$\mathbf{eig}(Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2} + (\epsilon - \kappa) \cdot \mathbf{I}) = \lambda_k + \epsilon - \kappa,$$

where λ_k is an eigenvalue, $k \in \{1, \dots, W\}$, W is the number of eigenvalues, $\mathbf{I}$ is the identity matrix and $\mathbf{eig}(\cdot)$ outputs eigenvalues of the input matrix [46].

Since κ is the smallest eigenvalue and $\boldsymbol{B}\boldsymbol{B}^T$ is not positive definite, we can conclude that $\kappa < 0$ and $\lambda_k \geq \kappa$ for k. It follows that $\lambda_k - \kappa + \epsilon > 0$, for all eigenvalues $\lambda_k, k \in \{1, \dots, W\}$. Hence, $Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2}$ is positive definite. Multiplying both sides of $Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2}$ by $Z^{1/2}$, we obtain that

$$\begin{aligned} & Z^{1/2}(Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2} + (\epsilon - \kappa) \cdot \mathbf{I})Z^{1/2} \\ & = \boldsymbol{B}\boldsymbol{B}^T + (\epsilon - \kappa) \cdot Z = \boldsymbol{B}\boldsymbol{B}^T + D. \end{aligned}$$

The determinant $|D| = (\epsilon - \kappa) \cdot Z > 0$ reveals that D is definite positive. Hence, we obtain $\boldsymbol{\omega} D \boldsymbol{\omega} > 0$ for all nonzero $\boldsymbol{\omega} \in$

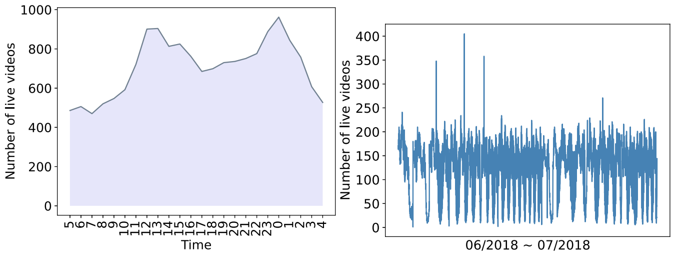

Fig. 6. Left: variations of live videos in a day; Right: variations of live videos during two months.

$\mathbb{R}^N$. Due to the positive definite matrix $Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2} + (\epsilon-\kappa)\cdot\mathbf{I}$, Finally, $\boldsymbol{H}$ is positive definite proved by the determinant value

$$|Z^{1/2}(Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2} + (\epsilon-\kappa)\cdot\mathbf{I})Z^{1/2}|$$
$$= |Z^{1/2}|\cdot|Z^{-1/2}\boldsymbol{B}\boldsymbol{B}^T Z^{-1/2} + (\epsilon-\kappa)\cdot\mathbf{I}|\cdot|Z^{1/2}| > 0.$$

□

Theorem 2 (Approximated Optimal Solution): If $\boldsymbol{\omega}^\star$ is the optimal solution of $\mathcal{P}_1$, one of the following is true:

a) $\sum_{i\in\mathcal{N}}\omega_i\cdot\nabla_O\mathcal{L}_i = 0$ and $\boldsymbol{\omega}$ leads to a Pareto stationary solution; and
b) $\sum_{i\in\mathcal{N}}\omega_i\cdot\nabla_O\mathcal{L}_i$ is a gradient descent direction maximising user i's utility function.

Proof: Based on Theorem 1, we substitute the term $\|\sum_i\omega_i\nabla_O\mathcal{L}_i\|_2^2$ by $\boldsymbol{B}\boldsymbol{B}^T + D$ and have that the improved upper bound is $\boldsymbol{\omega}\boldsymbol{H}\boldsymbol{\omega}$, where $\boldsymbol{H} = \boldsymbol{B}\boldsymbol{B}^T + D$. Then, we have that $\boldsymbol{\omega}^T\boldsymbol{H}\boldsymbol{\omega} = \boldsymbol{\omega}(\boldsymbol{B}\boldsymbol{B}^T + D)\boldsymbol{\omega} = \boldsymbol{\omega}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{\omega} + \boldsymbol{\omega}D\boldsymbol{\omega}$. Then, similar to [38], we can prove that statements (a) and (b) are feasible under the improved upper bound, i.e., $\boldsymbol{\omega}^T\boldsymbol{H}\boldsymbol{\omega}$. The **gap** between optimal and the approximated values is $\boldsymbol{\omega}D\boldsymbol{\omega} = \boldsymbol{\omega}^T\boldsymbol{H}\boldsymbol{\omega} - \boldsymbol{\omega}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{\omega}$. □

VI. Evaluation

A. Settings

We run our experiments on servers with an Intel Xeon Silver 4214 CPU@2.20 GHz and a Tesla P40 graphical card. For our GNN-based model, we set the learning rate of the shared and user decision models to 0.005 [38], [40], the attention vector $\boldsymbol{q}$'s dimension to 8, the number of attention heads to 2, ϵ to 1×10^{-6}, and the attention dropout to 0.5 based on experimental results and experience. We empirically set the dimension of each embedding of the migration path in O to 30. Two types of video streaming services were considered ($|\mathcal{K}| = 2$).

Dynamic Workload: Fig. 6 plots the number of simultaneous videos in the dataset of facebookvideoslive18 from 5 am to 4 am of the next day on Feb 4, 2018.[4] The number of video service invocations rises twice daily, from 07:00 to 12:00 noon and from 16:00 to midnight. It illustrates a particular pattern of users using video services in the time dimension, explaining the dynamic nature of the workload of the services in an edge network. We use the records of June and July in this dataset. We selected data records of live videos from Vietnam and the U.S. to synthesize the training dataset. Then we use a dataset synthesized from records in these regions on 09/02/2018 as a validation dataset.

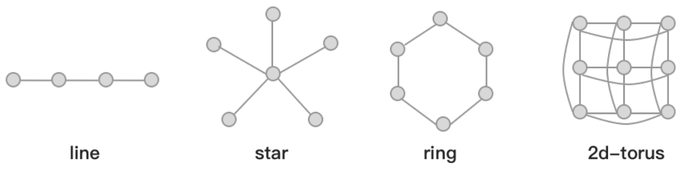

Fig. 7. Topologies of edge servers.

CPU Usage Versus Bandwidth Usage: Many Internet companies have proposed the Streaming Video Engine (SVE) [50] in response to the increasing demand for video services. When a stream from a source video delivers to an edge server in real-time, the edge server simultaneously transcodes and transsizes the video to users. We test the average CPU usage during video transcoding at a cloud server using FFmpeg.[5] As shown in Table II, we transcode and transsize an HD 2 k video to several different versions. We measured bitrates from 0.1 M/s to 3.7 M/s with 10 dimensions and resolutions from 192x144 to 1920x1080 with 6 dimensions. In the our evaluation, we normalize the values of all CPU usage.

Synthetic Dataset: First, we randomly combine the data in Table II with each piece of data in dataset facebookvideoslive18 to represent the bandwidth and computing resources demand of each video's service request. The dataset follows the form of a time series with several video requests at every time slot. Then, we randomly assign these requests to each user at each time slot. Therefore, each user generates a random bandwidth and computing resources workload following the dynamic characters of a realistic scenario.

Edge Network Topology: As shown in Fig. 7, we perform our evaluations with different network topologies of edge servers, including line, star, ring, and 2d-torus [51].

B. Baselines:

a) *Allocate by demand proportion (ADP):* ADP follows the proportional allocation principle which allocates resources proportionally based on the resource requests of all users.
b) *Fass [19]:* Fass ensures fairness in service request responses among multiple users under resource constraints. We use CVXOPT [49] to solve the minimization problem $\arg\min_{x_i^t} N^{-[\sum_{j\in\mathcal{M}} U_i^t(x_i^t)]}$ in each time slot for allocations under (21)–(22).
c) *MTP [18]:* MTP is designed explicitly for mobile edge scenarios and ensures Pareto fairness in resource allocation for multiple users. We adopt the multi-resource allocation strategy with maximum task product fairness approach from [18]. We solve $\max_{x_i^t}\sum_{i\in\mathcal{N}}\log x_i^t$ in each time slot for allocation using CVXOPT [49], under the constraints (21)–(22).

[4]https://sites.google.com/view/facebookvideoslive18/download

[5]https://ffmpeg.org/

TABLE II
CPU USAGE

CPU USAGE %		**BITRATES** Mbps									
		0.1M	0.5M	0.9M	1.3M	1.7M	2.1M	2.5M	2.9M	3.3M	3.7M
RESOLUTION	192x144	434.83	473.70	479.62	498.99	488.86	505.92	498.28	513.67	511.57	518.54
	320x240	455.23	527.48	534.66	561.34	574.38	563.94	582.79	578.62	597.91	587.54
	480x360	483.04	579.72	634.96	653.96	668.91	663.16	694.33	704.67	709.44	724.87
	640x480	488.81	610.58	665.72	720.12	731.34	784.56	792.97	815.50	834.83	817.12
	1280x720	578.95	774.62	901.16	988.83	1036.32	1086.05	1091.66	1125.35	1150.65	1118.29
	1920x1080	682.70	906.60	1061.29	1175.20	1221.72	1308.32	1349.10	1408.42	1407.08	1418.44

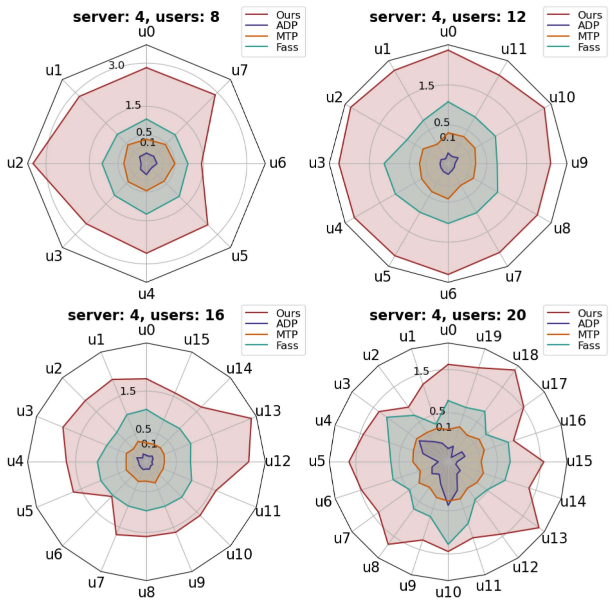

Fig. 8. Experiments with a fixed number of edge servers.

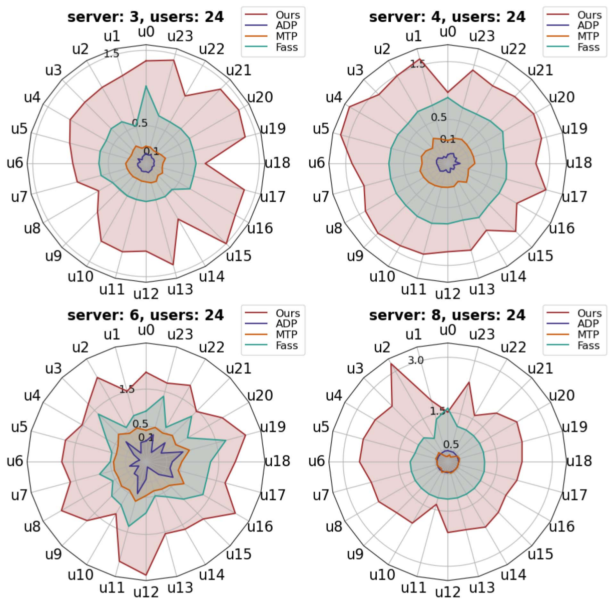

Fig. 9. Experiments with a fixed number of users.

C. Performance

Our experiments are conducted to address the following four research questions:

- **Q1:** How does our proposed algorithm perform under varying numbers of servers and users?
- **Q2:** What is the algorithm's performance under different network topologies?
- **Q3:** How effectively does the algorithm converge during training?
- **Q4:** To what extent does the algorithm's performance improve with the enhancements made to Lemma.1?

1) Performance Under Different Numbers of Users and Edge Servers: We start by conducting a performance evaluation of a line topology of edge servers. To achieve this, we randomly allocate the dynamic request of the number of live videos, as illustrated in Fig. 6 and Table II, to each user as their video stream request. This allocation is carried out over 8000 iterations, and subsequent evaluations are conducted over 100 rounds.

In Fig. 8, we initially set the number of edge servers to 4. Our proposed algorithm outperforms its peers for scenarios involving 8, 12, 15, and 20 users. Similarly, we fixed the number of users at 24 and show the results in Fig. 9. We vary the number of edge servers, noting that increasing the number of servers from 3 to 8 results in higher user utility obtained by the proposed one than its peers. hese two groups of experiments shown in Figs. 8 and 9 demonstrate that our proposed algorithm maintains its efficient performance guarantee even with changes in the number of users and servers. Our algorithm outperforms its peers in terms of user utilities. Specifically, when the number of servers is fixed, the utility of each user still surpasses its peers as the number of users increases. Thus, we provide an answer to **Q1** by demonstrating the performance of our proposed method even under varying numbers of users and servers.

2) Comparisons Under Different Network Topologies: We conduct experiments under three topologies, i.e., star, ring, and 2d-torus, shown in Fig. 7. Then, we take the data from the first three rows and first five columns of Table II as low-quality user requirements and the data from the last three rows and last five columns as high-quality user requirements. Additionally, to dive into the robustness of our proposed algorithm, we summarize three groups of workload distributions, I, II, and III, in which workload distribution I denote that about 50% of users request the high-quality streaming service and the remaining 50% choose low-quality, workload distribution II represents that 30% of users choose high-quality, and the other 70% select low-quality, and workload distribution III denotes that 70% choose high-quality, and the rest 30% choose low-quality. Table III illustrates that compared to its peers, the average column represents the average value of all user utilities, the max column represents

TABLE III
COMPARISONS UNDER DIFFERENT TOPOLOGIES WITH REQUEST DISTRIBUTIONS I, II, III

		average I, II, III	max I, II, III	min I, II, III	#best I, II, III	#lowest I, II, III
2d-torus	Ours	**1.321, 2.062, 2.137**	1.726, **2.463**, **2.691**	**0.666**, **1.611**, **0.6**	**23**, **23**, **21**	**0, 0, 0**
	ADP	0.176, 0.428, 0.474	**6.727**, 0.732, 0.847	-0.404, 0.008, 0.04	0, 0, 0	22, 24, 24
	MTP	0.198,0.745,0.817	0.305,0.856,0.961	0.03,0.493,0.563	0,0,0	2,0,0
	Fass	0.721,1.274,1.369	1.142,1.947,2.211	0.434,0.535,0.58	1,1,3	0,0,0
ring	Ours	**2.07**, **2.09**, **2.213**	**3.497,2.83,2.837**	**1.032,1.244**,-0.222	**18,18,17**	**0,0,1**
	ADP	0.393,0.404,0.405	0.753,0.71,0.756	0.027,0.007,0.044	0,0,0	18,18, 17
	MTP	0.68,0.673,0.673	0.848,0.804,0.824	0.467,0.468,0.469	0,0,0	0,0,0
	Fass	1.263,1.292,1.272	2.01,2.011,2.057	0.502,0.536,**0.552**	0,0,1	0,0,0
star	Ours	**1.893**, **1.876**, **2.026**	**2.532,2.315,3.067**	**1.472,1.388**,0.19	**16,17,16**	**0,0,0**
	ADP	0.389,0.388,0.39	0.748,0.632,0.749	-0.09, -0.056,-0.014	0,0,0	18,17,18
	Fass	0.672,0.684,0.678	0.786,0.821,0.894	0.391,0.42,0.386	0,1,0	0,0,0
	MTP	1.27,1.285,1.247	1.973,1.873,2.925	0.687,0.434,**0.675**	2,0,2	0,1,0

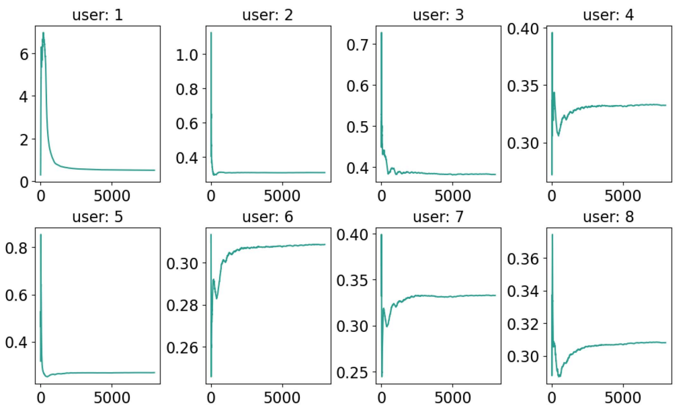

Fig. 10. User loss.

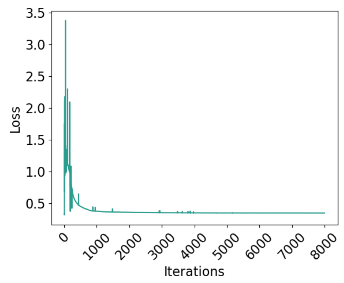

Fig. 11. Shared model's loss.

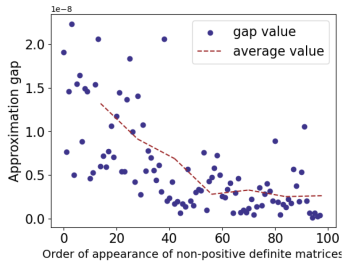

Fig. 12. Approximation gap.

the maximum value of all user utilities, and the min column represents the minimum value of all user utilities. we observed that our proposed algorithm maintains consistent advantages in terms of average, max, and min values across different topological structures compared to the comparison algorithm. Moreover, in terms of #best and #lowest, our proposed algorithm can guarantee that most users get better utility than its peers. Our algorithm exhibits significant superiority over them, even in scenarios where network topologies possess more intricate connections. Specifically, when considering the 2d-torus topology with task request distributions I–III, our algorithm yields an advantage in performance for up to 23, 23, and 21 users out of 24. According to the statistics presented in Table III, our algorithm provides the best utility for approximately 93%, 98%, and 92% of users under 2d-torus, ring, and star topologies, respectively, compared to its peers. Thus, we provide an answer to **Q2** by demonstrating the performance of our proposed method under different network topologies.

3) Convergence of the Learning Algorithm: According to the experiment on four servers and eight users in Fig. 8, we show the convergence process of its loss functions in Figs. 10 and 11. After many iterations, our proposed algorithm ultimately achieves convergence. Fig. 11 depicts the convergence of the shared model, with the loss fluctuation dissipating after approximately 5000 iterations. The user loss in Fig. 10 illustrates the learning process of each user within the algorithm, and the loss function of each user converges to a specific fixed value. Similar to the shared model, the trend toward convergence is consistent across all users.

Furthermore, Fig. 12 illustrates the gap $\omega D \omega$, as proven by Theorem 1. The red dotted line in Fig. 12 represents the average of the approximated gap, which converges along with training iterations. As a result, we indirectly confirm the convergence of the loss function in Fig. 11 by examining the value of the approximated gap. Thus, we provide an answer to **Q3** by demonstrating the convergence of our proposed method.

4) Algorithm Stability: To assess the efficacy of the evaluation process, we perform evaluations on the trained algorithms separately, starting from 100 rounds up to 400 rounds in increments of 50, and subsequently calculate the average results, as illustrated in Fig. 13, with eight users and four servers. The experimental result indicates that our algorithm consistently maintains superior performance for each user, regardless of the duration, thereby demonstrating the stability of our proposed algorithm.

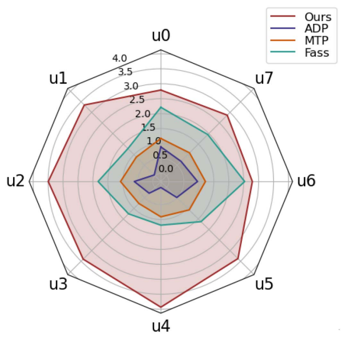

Fig. 13. Average utilities.

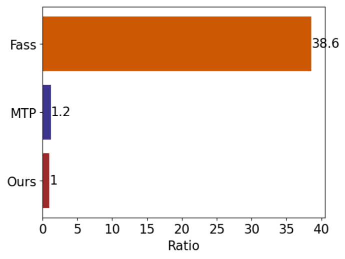

Fig. 14. Decision making latency.

5) Latency of Decision Making: Fig. 14 displays the algorithm's latency during evaluation, with our algorithm serving as the scale base. Due to the considerable number of dimensions and constraints in the Fass algorithm, the CVXOPT module requires a significant amount of time to implement the interior point method, resulting in a much longer execution time than MTP and our algorithm. Besides, our algorithm exhibits an obvious advantage over MTP, with an execution speed increased by approximately 20%.

6) Effectiveness of Lemma 1: Actually, when $\{\nabla_O \mathcal{L}_i\}$ satisfies $\mathcal{S}_1$, an alternative approach is to initialize $\hat{i} = 1, \forall \hat{i} \in \hat{\mathcal{N}}$ and $\tilde{i} = 0, \forall \tilde{i} \in \tilde{N}$ in Lemma 1, which we refer to as a zero-gradient approach. This approach represents a feasible solution that satisfies (21)–(22) and reduces the algorithm's complexity due to omitting the procedure of Algorithm 4. However, this approach can impact the learning accuracy since the gradient update becomes invalid, rendering $\boldsymbol{\mathcal{P}_1}$ equivalent to 0 too early. We consider a utility vector for users under different baselines denoted by $\boldsymbol{U}_{ADP}, \boldsymbol{U}_{Fass}$ and $\boldsymbol{U}_{MTP}$. We define $p = \sum_{i \in \mathcal{N}} \mathbb{I}(\boldsymbol{U}(i) - \boldsymbol{U}_x(i))/3N$, where $x \in \{ADP, Fass, MTP\}$, and $\mathbb{I}(Y) = 1$ if $Y \geq 0$, otherwise $\mathbb{I}(Y) = 0$. The numerator in p accumulates the indicator function, where the variable represents the difference between the utility of all users under our proposed algorithm and the utility under the comparison algorithm. It indicates the total number of users whose utility values are higher than the comparison algorithms. We employ the findings from Section VI-C1 to contrast our algorithm with the zero-gradient scheme for all users under all baselines. Fig. 15 illustrates that the p values of our proposed algorithm are 100% in each figure, while the corresponding values of the zero-gradient scheme are 91.7%, 88.9%, 69.4%, and 55.6%. These experimental results

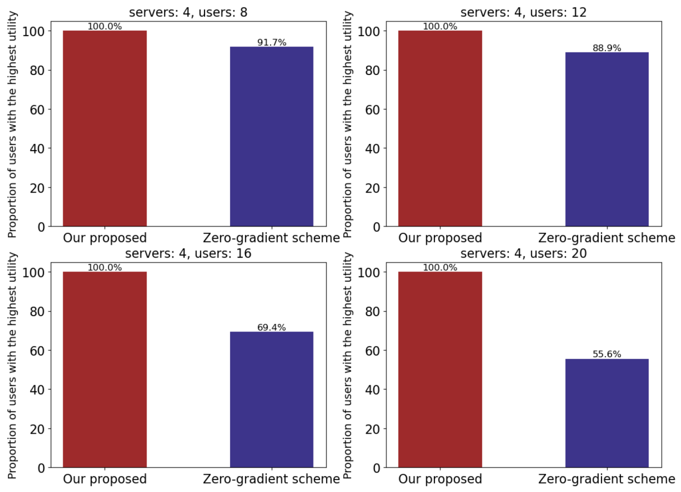

Fig. 15. Ours v.s Zero-gradient scheme.

demonstrate that our algorithm performs better than the zero-gradient scheme for all users, which confirms the effectiveness and robustness of our method under Lemma 1. Thus, we provide an answer to **Q4** by demonstrating the improvement of our proposed method.

VII. Conclusion

As edge computing faces challenges in collaborative service provision due to limited and unevenly distributed resources, we propose a resource allocation algorithm to guarantee the fairness among all users. By leveraging GNN to model graph embeddings, our proposed scheme does so with approximate Pareto optimality. Simulation results demonstrate its superiority over existing fairness-ensuring algorithms regarding QoE for all users. While our work has primarily focused on designing aggregation techniques for multiple loss functions during the gradient descent stage, our proposed algorithm can be extended to address the fairness problem when some users have minimum QoE requirements. Future research should also explore a fairness-ensuring scheme that employs gradient aggregation methods to find the Pareto-optimal frontier for multiple Pareto-optimal solutions.

References

[1] F. Loh, F. Wamser, F. Poignée, S. Geißler, and T. Hoßfeld, "Youtube dataset on mobile streaming for internet traffic modeling and streaming analysis," *Sci. Data*, vol. 9, no. 1, pp. 1–12, 2022.

[2] L. F. de Souza Cardoso, F. C. M. Q. Mariano, and E. R. Zorzal, "A survey of industrial augmented reality," *Comput. Ind. Eng.*, vol. 139, 2020, Art. no. 106159.

[3] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[4] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, "Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm," *Sensors*, vol. 21, no. 3, 2021, Art. no. 779.

[5] X. Zhu and M. Zhou, "Multiobjective optimized cloudlet deployment and task offloading for mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15 582–15 595, Oct. 2021.

[6] F. Fu, Y. Kang, Z. Zhang, and F. R. Yu, "Transcoding for live streaming-based on vehicular fog computing: An actor-critic DRL approach," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 1015–1020.

[7] K. Park and M. Kim, "EVSO: Environment-aware video streaming optimization of power consumption," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 973–981.

[8] C. Ge and N. Wang, "Real-time QoE estimation of DASH-based mobile video applications through edge computing," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 766–771.

[9] X. Chen et al., "A universal transcoding and transmission method for livecast with networked multi-agent reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[10] R. Viola, A. Martin, M. Zorrilla, and J. Montalbán, "MEC proxy for efficient cache and reliable multi-CDN video distribution," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast.*, 2018, pp. 1–7.

[11] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Decentralized resource allocation for video transcoding and delivery in blockchain-based system with mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 68, no. 11, pp. 11 169–11 185, Nov. 2019.

[12] P. Lai et al., "QoE-aware user allocation in edge computing systems with dynamic QoS," *Future Gener. Comput. Syst.*, vol. 112, pp. 684–694, 2020.

[13] C. Zhang, J. Liu, Z. Wang, and L. Sun, "Look ahead at the first-mile in livecast with crowdsourced highlight prediction," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1143–1152.

[14] Q. He et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[15] Y.-H. Hung, C.-Y. Wang, and R.-H. Hwang, "Combinatorial clock auction for live video streaming in mobile edge computing," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 196–201.

[16] M. Ashour, J. Wang, C. Lagoa, N. Aybat, and H. Che, "Non-concave network utility maximization: A distributed optimization approach," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[17] L. Chunlin, T. Jianhang, and L. Youlong, "Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud," *Cluster Comput.*, vol. 21, no. 2, pp. 1331–1348, 2018.

[18] E. Meskar and B. Liang, "Fair multi-resource allocation with external resource for mobile edge computing," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 184–189.

[19] S. Li, J. Huang, B. Cheng, L. Cui, and Y. Shi, "FASS: A fairness-aware approach for concurrent service selection with constraints," in *Proc. IEEE Int. Conf. Web Serv.*, 2019, pp. 255–259.

[20] Y. Han, S. Shen, X. Wang, S. Wang, and V. C. Leung, "Tailored learning-based scheduling for kubernetes-oriented edge-cloud system," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[21] A. Rafiq et al., "Knowledge defined networks on the edge for service function chaining and reactive traffic steering," *Cluster Comput.*, vol. 26, pp. 613–634, 2022.

[22] H. Zhu, V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin, "Network planning with deep reinforcement learning," in *Proc. ACM SIGCOMM Conf.*, 2021, pp. 258–271.

[23] D. Pujol-Perich et al., "IGNNITION: Fast prototyping of graph neural networks for communication networks," in *Proc. SIGCOMM'21 Poster Demo Sessions*, 2021, pp. 71–73.

[24] O. Hope and E. Yoneki, "GDDR: GNN-based data-driven routing," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 517–527.

[25] X. Lin et al., "GSO-Simulcast: Global stream orchestration in simulcast video conferencing systems," in *Proc. ACM SIGCOMM Conf.*, 2022, pp. 826–839.

[26] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[27] Y. Sun et al., "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[28] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 13 065–13 076, Aug. 2021.

[29] Z. Duan et al., "A novel load balancing scheme for mobile edge computing," *J. Syst. Softw.*, vol. 186, 2022, Art. no. 111195.

[30] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1941–1954, Fourt Quarter 2022.

[31] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.

[32] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[33] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2287–2295.

[34] X. Xiao et al., "Novel workload-aware approach to mobile user reallocation in crowded mobile edge computing environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8846–8856, Jul. 2022.

[35] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2021.

[36] F. Wang et al., "DeepCast: Towards personalized QoE for edge-assisted crowdcast with deep reinforcement learning," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1255–1268, Jun. 2020.

[37] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5/6, pp. 313–318, 2012.

[38] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 525–536.

[39] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 950–958.

[40] X. Wang et al., "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032.

[41] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv: 1710.10903*.

[42] N. Yang, Z. Zheng, M. Zhou, X. Guo, L. Qi, and T. Wang, "A domain-guided noise-optimization-based inversion method for facial image manipulation," *IEEE Trans. Image Process.*, vol. 30, pp. 6198–6211, 2021.

[43] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[44] Z. Huang, X. Xu, H. Zhu, and M. Zhou, "An efficient group recommendation model with multiattention-based neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4461–4474, Nov. 2020.

[45] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, "Qualitatively characterizing neural network optimization problems," 2014, *arXiv:1412.6544*.

[46] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[47] N. J. Higham, *A Survey of Componentwise Perturbation Theory*, vol. 48. Providence, RI, USA: American Mathematical Society, 1994.

[48] N. Jorge and J. W. Stephen, "Numerical optimization," Springer, 2006.

[49] M. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: A python package for convex optimization," Version 1.1.6, 2013. [Online]. Available: https://cvxopt.org

[50] M. A. Salehi, "Cloud-based interactive video streaming service," in *Proc. 10th Int. Conf. Utility Cloud Comput.*, 2017, pp. 183–184.

[51] Z. Zhong et al., "P-FedAvg: Parallelizing federated learning with theoretical guarantees," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

Cheng Zhang received the MS degree in electrical engineering from Zhejiang University, China in 2013, Currently, he is working towards the PhD degree in computer science and technology with Zhejiang University. His research interests include edge computing and edge intelligence.

Jianwei Yin received the PhD degree in computer science from Zhejiang University (ZJU) in 2001. He was a visiting scholar with the Georgia Institute of Technology. He is currently a full professor with the College of Computer Science, ZJU. Up to now, he has published more than 100 papers in top international journals and conferences. His current research interests include service computing and business process management. He is an associate editor of *IEEE Transactions on Services Computing*.

Shuiguang Deng (Senior Member, IEEE), received the BS and PhD degrees in computer science from the College of Computer Science and Technology in Zhejiang University, China, in 2002 and 2007, respectively. He is currently a full professor with the College of Computer Science and Technology in Zhejiang University, China. He previously worked with the Massachusetts Institute of Technology in 2014 and Stanford University in 2015 as a visiting scholar. His research interests include edge computing, service computing, cloud computing, and business process management. He serves for the journal *IEEE Transactions on Services Computing, Knowledge and Information Systems, Computing*, and *IET Cyber-Physical Systems: Theory and Applications* as an associate editor. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a fellow of IET.