

Enhanced Profit-Driven Optimization for Flexible Server Deployment and Service Placement in Multi-User Mobile Edge Computing Systems

Juan Fang ¹, Member, IEEE, Shen Wu ¹, Student Member, IEEE, Shuaibing Lu ¹, Member, IEEE, Ziyi Teng ¹, Student Member, IEEE, Huijie Chen ¹, Member, IEEE, and Neal N. Xiong ², Senior Member, IEEE

Abstract—Edge computing has emerged as a promising paradigm to meet the increasing demands of latency-sensitive and computationally intensive applications. In this context, efficient server deployment and service placement are crucial for optimizing performance and increasing platform profit. This paper investigates the problem of server deployment and service placement in a multi-user scenario, aiming to enhance the profit of Mobile Network Operators while considering constraints related to distance thresholds, resource limitations, and connectivity requirements. We demonstrate that this problem is NP-hard. To address it, we propose a two-stage method to decouple the problem. In stage I, server deployment is formulated as a combinatorial optimization problem within the framework of a Markov Decision Process (MDP). We introduce the Server Deployment with Q-learning (SDQ) algorithm to establish a relatively stable server deployment strategy. In stage II, service placement is formulated as a constrained Integer Nonlinear Programming (INLP) problem. We present the Service Placement with Interior Barrier Method (SPIB) and Tree-based Branch-and-Bound (TDB) algorithms and theoretically prove their feasibility. For scenarios where the number of users changes dynamically, we propose the Distance-and-Utilization Balance Algorithm (DUBA). Extensive experiments validate the exceptional performance of our proposed algorithms in enhancing the profit.

Index Terms—Mobile edge computing, profit-driven optimization, reinforcement learning, integer nonlinear programming.

I. INTRODUCTION

WITH the increasing number of mobile users, even though mobile devices are becoming smarter and more capable, the emergence of new applications and the growing demand for portable services are bringing significant computational and communication burdens. In this context, Mobile Edge Computing (MEC), as an innovative computing paradigm, offloads

computation-intensive tasks from resource-constrained mobile devices to edge servers located closer to end users. This effectively alleviates the computational load on mobile devices and the core network, while addressing the issue of high-latency communication. However, the expensive cost and limited computational capacity of these servers, along with the instability of activities among multiple end-users, present a series of challenges for enhancing the profit of Mobile Network Operators (MNOs).

A. Motivation and Challenges

In MEC, edge server placement and service placement are coupled, which is crucial for MEC systems. Existing research inevitably considers these factors, but the emphasis on edge server placement and service placement varies in different studies, and few studies consider the two together.

In terms of server deployment, existing research [1] introduced a two-stage algorithm utilizing binary and genetic algorithms. However, two-stage algorithms typically demand substantial computing resources. Moreover, the algorithm may lack adaptability to dynamic environments. On the other hand, research [2] applies reinforcement learning to minimize network latency and the number of edge servers. However, this model does not consider server heterogeneity. In this case, in order to fully utilize the resources of edge servers and provide better services, it is not only necessary to consider the heterogeneity of servers, but also to make appropriate service placement on the basis of existing server placement schemes.

In terms of service placement, many existing studies skip edge server placement, assuming that edge servers are pre-deployed at base stations [3], [4], [5], [6], [7], [8]. In such cases, the connection between edge server deployment and service placement is severed. Besides, this assumption is inappropriate in many cases because the major function of the base station is different from that of the edge server. In addition, the overall profit of the MEC platform is also a key factor to consider [9]. Telecom operators may find it challenging to bear the cost of deploying edge servers for each base station. Although [10] proposed a solution for joint edge server deployment and service placement with the goal of maximizing the profit of MNOs. However, server acquisition cost were not fully considered in its modeling process. In addition, the clustering algorithm used

Received 19 January 2024; revised 8 September 2024; accepted 6 October 2024. Date of publication 9 October 2024; date of current version 15 November 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62202019 and Grant 61202076, and in part by Beijing Municipal Natural Science Foundation under Grant 4192007, along with other government sponsors. Recommended for acceptance by Dr. Mugen Peng. (Corresponding author: Juan Fang.)

Juan Fang, Shen Wu, Shuaibing Lu, Ziyi Teng, and Huijie Chen are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: fangjuan@bjut.edu.cn; wushen@emails.bjut.edu.cn; lshuaibing@bjut.edu.cn; tengziyi@emails.bjut.edu.cn; chenhuijie@bjut.edu.cn).

Neal N. Xiong is with the Department of Computer, Mathematical and Physical Sciences, Sul Ross State University, Alpine, TX 79830 USA (e-mail: xiongnaxue@gmail.com).

Digital Object Identifier 10.1109/TNSE.2024.3477453

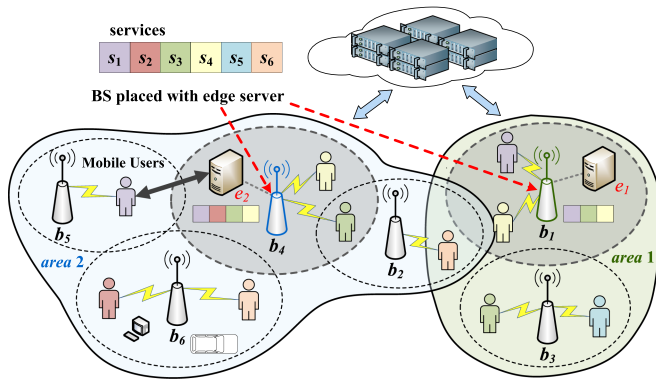


Fig. 1. Joint server deployment and service placement in a multi-user scenario.

in [10] makes it difficult to directly integrate real-time indicators and pure interior point algorithms have some limitations when solving integer programming problems.

In order to maximize the profit of MNOs and adapt to the ever-evolving demands of mobile networks, this paper delves deep into the study of joint server deployment and service placement in multi-user scenarios. To clarify this problem, we turn to the example provided in Fig. 1. Within this illustration, six base stations are depicted, with servers positioned at b_1 and b_4 . Represented by distinct-colored blocks (s_1 to s_6), these services encompass varied computational tasks, while users of the same color denote corresponding service requests. Some services have already been placed on edge servers. Each service replica serves a single user and generates income. If there is no matching service replica within the server, the request will be forwarded to the cloud. In region 2, servers e_2 could deal with the service requests from users associated with b_2 , b_4 , b_5 , and b_6 . However, when dealing with service requests from b_2 , b_5 , and b_6 , additional transmission cost is incurred, thus giving rise to some issues and challenges:

- i) How to determine the number, and deployment locations of servers which can be able to serve as many users as possible in various areas considering the cost of MNOs. In Fig. 1, deploying only one server incurs the lowest cost but does not cover all users. On the other hand, deploying servers in all six base stations is cost-prohibitive. Furthermore, when deploying two servers as depicted in Fig. 1, namely e_1 and e_2 , relocating the deployment position of e_2 from b_4 to b_5 in area 2 will significantly increase the transmission cost. Hence, determining the number and deployment locations of servers is of paramount importance. After determining the number and deployment locations of servers, we must also accurately determine the specifications of the servers, including processing power and storage capacity. This is because servers with more processing power and storage capacity cost more.
- ii) How to determine service placement strategies within limited server capacity to increase the profit of MNOs. Taking e_2 as an example, our observations indicate that the service request rate for s_6 is exceptionally high. However, there are no copies of s_6 placed on the server. This suggests

that from the perspective of the MNOs, the potential profit generated by providing this service may be relatively limited. Service placement considers not only the service request rate but also its price, under the constraints of limited storage and computing capacity of the edge server

B. Contributions and Paper Organization

In this paper, we investigate the problem of server deployment and service placement in a multi-user scenario, aiming to enhance the profit of MNOs while considering constraints related to distance thresholds, resource limitations, and connectivity requirements. Our research contributions are summarized as follows:

- We formulate a profit model to address the challenges of server deployment and service placement in multi-user scenarios. To better tackle this problem, we employ a two-stage method to decouple the problem. We prove that the problem of deploying servers to maximize profit under the constraints is an NP-hard problem.
- We formalize the server deployment scheme as a Combinatorial Optimization Problem (COP) in stage I. The problem is formulated as an MDP to efficiently describe the state space, action space, and penalty function. Subsequently, we focus on the internal service placement strategy in stage II, which builds upon the deployment results of edge servers. We reduce this issue to a constrained INLP problem. Then, we propose a strategy to find the integer optimal solution and theoretically prove the feasibility. Additionally, we introduce the Distance-and-Utilization Balance Algorithm to strategize the service placement for new users.
- We performed thorough experiments to assess the effectiveness of our approach, contrasting it with various baseline methods using a genuine base station dataset supplied by Shanghai Telecom. Our extensive experimental findings consistently affirm our proposed algorithm's superior efficacy.

The remainder of this paper is organized as follows: Section II provides an overview of relevant research. In Section III, we modeled the problem. Section IV realizes the problem's decoupling. Section V discusses the proposed solution to this problem, while Section VI illustrates the comparative experimental evaluation results. Finally, in Section VII, we conclude the paper.

II. RELATED WORK

Current researches on server deployment and service placement predominantly focus on computing latency, cost, or energy consumption, overlooking the profit requirements from the service provider's perspective.

A. Server Deployment

Ning et al. [11] formulated models for dynamic multi-user computation offloading and edge server deployment problems using stochastic game theory. However, the applicability is limited despite considering unmanned aerial vehicle scenarios. Li et al. [12] introduced the concept of 5G User Plane

Function (UPF) for calculating access latency and designed a particle swarm optimization-based algorithm to optimize profit. Guo et al. [13] introduced a method employing k-means and mixed-integer quadratic programming to distribute workload evenly among edge clouds. Jia M et al. [14] studied cloud placement and assignment of mobile users to clouds. Wang et al. [15] used mixed integer programming to address the multi-objective constrained problem of placing edge servers. Nguyen et al. [16] introduced a method using deep reinforcement learning to address the non-convex problem of reducing computational cost. Kasi et al. [17] applied genetic algorithm and local search algorithm to strategically place edge servers. Xu et al. [18] developed a collaborative approach for server quantification and placement using clustering algorithms. Ling et al. [19] combined the server placement problem with graph convolutional networks to optimize network latency and energy consumption. Qu et al. [20] focused on finding a strategy for server placement that maximizes the expected workload under uncertain edge server failures. Xu et al. [21] studied the joint deployment and allocation of virtual network functions, aiming to achieve a balance between cost and Quality of Service (QoS).

B. Service Placement

Han et al. [22] addressed energy-saving and flexible service layout in mobile edge cloud networks supporting cell scaling. Sami et al. [23] proposed a deep reinforcement learning solution for fast and proactive service updates to meet user needs. Li et al. [24] proposed a collaborative placement and scheduling framework for joint optimization design. Gao et al. [25] designed a two-step method to explore the simultaneous optimization of MEC access network selection and service arrangement. Shen et al. [26] propose a fully Bayesian processing recommendation framework, which can easily and effectively integrate any type of auxiliary information for project recommendation. Gao et al. [27] introduced a method based on iteration to tackle the issue of simultaneously optimizing MEC access network selection and service placement. Yang et al. [28] proposed a new fingerprint recognition scheme based on a set of combined invariant moments (geometric moments and Zernike moments) features to ensure communication security. Fang et al. [29] created a mapping scheme and system power consumption between application modules and basic resource devices. Li et al. [30] jointly optimized the graph task allocation and resource allocation of the MEC system. Teng et al. [31] studied joint active caching and cache replacement strategies in the MEC network.

Qu et al. [32] developed a parameter-sharing model caching scheme that improved storage efficiency by allowing AI models to share parameter blocks. Wu et al. [33] presented a hybrid runtime framework for managing multiple Deep neural networks (DNNs) at the edge, optimizing throughput and improving efficiency. Chen et al. [34] proposed an admission control mechanism for time-sensitive edge services, allowing providers to decide on accepting or redirecting requests to maximize revenue and ensure QoS. Jin et al. [35] modeled the virtual network function (VNF) deployment problem as a mixed integer linear programming (MILP) to minimize resource consumption. Li

et al. [36] proposed Finedge which empirically allocated the optimal CPU core and cost-effective CPU quota based on QoS requirements. Chen et al. [37] proposed a data-driven cooperative task allocation (DCTA) method to address task allocation in multi-task transfer learning. Pan et al. [38] proposed a method to jointly optimize container caching and request distribution, revealing a joint optimization problem based on the ski rental problem.

A closely related work to ours is [10]. [10] proposed a solution for joint edge server deployment and service placement. Compared to [10], our work has the following advantages and improvements: 1) Our study models the entire model in more details, and takes into account the acquisition cost of the server itself. 2) The clustering algorithm used in [10] is difficult to directly integrate real-time indicators such as latency, so we use reinforcement learning for server deployment. 3) We introduced the branch-and-bound method for improvement and theoretically proved the feasibility. 4) We also designed new algorithms to establish flexible connectivity with base stations and servers when the number of users changes.

In this paper, we jointly optimize the server deployment and service placement. Our objective is to maximize the profit of MNOs while taking into account various constraints that include distance thresholds, resource limitations, and connectivity requirements.

III. SYSTEM MODELS

A. The MEC Environment

Several key notations and their meanings have been succinctly summarized in Table I.

1) *Cloud*: The remote cloud is positioned at a considerable distance from all edge servers, denoted as R . It is assumed that the remote cloud hosts a diverse range of services and possesses the capability to process all service requests forwarded from the edge servers. Here, we use $S = \{s_w\}$ to denote the set of services, where s_w stands for a specific service. The volume of each service is denoted as z_s . Based on historical service request records from users, the average request rate of s_w can be denoted as ξ_{s_w} .

2) *Edge Layer*: We use B denotes the set of base stations, where $B = \{b_k\}$. The position of b_k is given by L_{b_k} . Furthermore, we introduce the notion of neighbor vectors for the base station b_k , denoted as $B_{(b_k)}$. The neighbor vectors of b_k consist of those other base stations whose distance from b_k is less than the threshold D_{th} . These adjacent base stations are capable of communicating with each other. E denotes the set of deployed edge servers, where $E = \{e_j\}$. Here, e_j manages multiple base stations, with storage capacity Z_{e_j} and average service rate μ_{e_j} . Different services could hold varying numbers of replicas on the edge servers, denoted as $M_{e_j}^{s_w}$. Each replica serves a single service request. G represents the set of server specifications, encompassing processing power, and storage capabilities. We define a two-dimensional matrix $\Delta(|E| \times |B|)$ to represent the connectivity between base stations and servers. The binary variable $\Delta_{e_j, b_k} \in \{0, 1\}$ is used to denote whether the edge server e_j is connected to the base station b_k .

TABLE I
SUMMARY OF SYMBOLS IN THE SYSTEM MODE

Symbol	Definition
R	Remote cloud
B	Set of base stations, $B = \{b_k\}$
L_{b_k}	Location of b_k
$\tilde{B}_{(b_k)}$	Neighbor vectors of b_k
E	Set of edge servers, $E = \{e_j\}$
G	Set of server specifications
Δ	Two-dimensional matrix to represent the connectivity
Δ_{e_j, b_k}	Boolean variable, whether e_j is connected to b_k
D_{th}	Distance threshold
\mathbb{D}_{e_j}	Average distance within the cluster of e_j
$d_{(b_k, e_j)}$	Distance between b_k and e_j
Z_{e_j}	Storage capacity of e_j
ρ_{e_j}	Utilization of e_j
μ_{e_j}	Average service rate of e_j
C_{e_j}	Cost of e_j from providing services
I_{e_j}	Income of e_j from providing services
O_{e_j}	Profit of e_j from providing services
S	Set of services, $S = \{s_w\}$
$S_{(e_j)}$	Set of services placed inside e_j
z_{s_w}	Volume of s_w
ξ_{s_w}	Average request rate of s_w
i_{s_w}	Income of providing s_w
σ_{s_w}	Unit transmission delay cost for s_w
λ_{s_w}	Unit computation delay cost for s_w
U	Set of users, $U = \{u_v\}$
$U_{(b_k/e_j)}$	Set of users connected to b_k/e_j
$N_{(b_k)}^{s_w}$	Number of requests for s_w received by b_k .
w_{b_k}	Number of all requests received by b_k
$N_{(e_j)}^{s_w}$	Number of requests for s_w received by e_j .
$N_{(R)}^{s_w}$	Number of requests for s_w forwarded to R .
$M_{(e_j)}^{s_w}$	Number of s_w placed on e_j

3) *End Layer*: Let U to denote the set of users, where $U = \{u_v\}$. We use $U_{(b_k)}$ to represent the set of users connected to b_k , where $U_{(b_k)} = \{u_v \rightarrow b_k | u_v \in U\}$. We use $U_{(e_j)}$ to denote the set of users that connected to e_j , where $U_{(e_j)} = \{u_v \rightarrow e_j | u_v \in U\}$. The estimated number of requests for service s_w received by b_k can be approximated as: $N_{(b_k)}^{s_w} = |U_{(b_k)}| \times \xi_{s_w}$. The number of requests for s_w received by e_j is given by: $N_{(e_j)}^{s_w} = \sum_{b_k \in B} N_{(b_k)}^{s_w} \cdot \Delta_{e_j, b_k}$.

B. Profit Model

We advocate considering the profit of MNOs, which are comprised of the cost and income. We use C_{e_j} to denote the cost of e_j , composed of the servers' acquisition cost, transmission cost, and computation cost. I_{e_j} is the income of e_j collected from providing services to users.

1) *Server Acquisition Cost*: Server acquisition cost includes purchasing the necessary server hardware. We assume that the servers are heterogeneous, and different prices are determined by varying processors and storage, denoted as $C_{e_j}^{alloc}$.

2) *Transmission Cost*: The total transmission cost is divided into two components: one is the transmission cost generated by servers dealing with service requests, and the other is the transmission cost generated by cloud server dealing with service requests.

As one edge server could cover several base stations, we use the average distance between these base stations and the server to measure how far the service requests from these stations need to be sent to reach the edge server. The average distance of e_j is defined as: $\mathbb{D}_{e_j} = \sum_{b_k \in B} D_{(b_k, e_j)} \cdot \Delta_{e_j, b_k} / \sum_{b_k \in B} \Delta_{e_j, b_k}$. Transmission cost per unit distance for s_w is denoted as σ_{s_w} . The overall transmission cost of all service requests to e_j is formulated as: $C_{(b_k, e_j)}^{trans} = \sum_{s_w \in S_{(e_j)}} \mathbb{D}_{e_j} \cdot \sigma_{s_w} \cdot N_{e_j}^{s_w}$.

When the server does not have extra space to handle new service requests, the cloud needs to handle these redundant requests. The number of service requests for s_w forwarded to the remote cloud is given by $N_{(R)}^{s_w} = N_{(e_j)}^{s_w} - \min\{N_{(e_j)}^{s_w}, M_{(e_j)}^{s_w}\}$, where $\min\{N_{(e_j)}^{s_w}, M_{(e_j)}^{s_w}\}$ is the number of requests actually processed by e_j . The distance between e_j and R is denoted as $D_{(e_j, R)}$. Thus, the cost of transmitting all redundant service requests from e_j to R can be formulated as $C_{(e_j, R)}^{trans} = \sum_{s_w \in S_{(e_j)}} D_{(e_j, R)} \cdot \sigma_{s_w} \cdot N_{(R)}^{s_w}$. So the total transmission cost is defined as $C^{trans} = C_{(b_k, e_j)}^{trans} + C_{(e_j, R)}^{trans}$.

3) *Computation Cost*: The computation cost is incurred due to computation delay. In order to better represent the computation delay, the processing of service requests by each edge server is modeled as an M/M/1 queuing model. Therefore, according to the definition of the queuing model, the computation delay of e_j in processing service requests can be derived as: $\mathbb{T}_{e_j}^{comp} = 1 / (\mu_{e_j} - \sum_{s_w \in S_{(e_j)}} \min\{N_{e_j}^{s_w}, M_{e_j}^{s_w}\} + 1)$. Let λ_{s_w} represent the unit computation delay cost for s_w . Then, the computation delay cost for all service requests sent to e_j can be modeled as: $C_{e_j}^{comp} = \sum_{s_w \in S_{(e_j)}} \mathbb{T}_{e_j}^{comp} \cdot \lambda_{s_w} \cdot \min\{N_{e_j}^{s_w}, M_{e_j}^{s_w}\}$.

4) *Income*: Income is collected by providing services to users. Let us assume that a replica of s_w serves a request of s_w and generates income i_{s_w} . Consequently, the income of e_j can be expressed as: $I_{e_j} = \sum_{s_w \in S_{(e_j)}} \min\{N_{e_j}^{s_w}, M_{e_j}^{s_w}\} \cdot i_{s_w}$. The cost for services provided by e_j is defined as follows:

$$C_{e_j} = C_{e_j}^{alloc} + C_{e_j}^{trans} + C_{e_j}^{comp}. \quad (1)$$

Intuitively, the profit of e_j can be expressed as:

$$O_{e_j} = I_{e_j} - C_{e_j}. \quad (2)$$

Our overall objective is to maximize the profit of all servers.

IV. PROBLEM DEFINITION

In this paper, we investigate the server deployment and service problem. Our objective is to maximize the profit of all servers under constraints including distance thresholds, resource limitations, and connectivity requirements. Thus, the formal definition of this problem is as follows:

$$\mathbf{P}_1 : \text{maximize } \sum_{e_j \in E} O_{e_j}(\Delta, S_{(e_j)}), \quad (3)$$

$$\text{s.t. } \sum_{e_j \in E} \Delta_{e_j, b_k} = 1, \quad \forall b_k \in B, \quad (4)$$

$$\Delta_{e_j, b_k} \cdot d_{(e_j, b_k)} \leq D_{th}, \quad \forall e_j \in E, \quad \forall b_k \in B, \quad (5)$$

$$\sum_{s_w \in S(e_j)} M_{e_j}^{s_w} \cdot z_{s_w} \leq Z_{e_j}, \quad \forall e_j \in E, \quad (6)$$

$$M_{e_j}^{s_w} \geq 0, \quad \forall e_j \in E, \quad \forall s_w \in S(e_j), \quad (7)$$

$$\sum_{s_w \in S(e_j)} \min \{N_{e_j}^{s_w}, M_{e_j}^{s_w}\} \leq \mu_{e_j}, \quad \forall e_j \in E. \quad (8)$$

\mathbf{P}_1 is the objective function. Equations (4) to (8) are the constraints. Equation (4) stipulates that each base station connects to a single-edge server. Equation (5) determines that the distance between a base station and the connected server must not exceed the preset threshold D_{th} . Equation (6) emphasizes that the total volume of service replicas on each server must not exceed the respective capacities. Equation (7) asserts that non-negative replicas on servers. Equation (8) indicates that the number of service requests processed by the server cannot exceed its processing rate. The solution space for server deployment and service placement problems is very huge. We introduce a strategy named the two-stage method to achieve problem decoupling.

A. Stage I

In stage I, we isolate the portion $C_{e_j}^{alloc} + C^{trans}$ from \mathbf{P}_1 for separate analysis, considering it as the objective of stage I. This phase presents a balancing act, as reducing $C_{e_j}^{alloc}$ results in an increase of C^{trans} and vice versa. Therefore, we formulate a COP by minimizing the combined cost. It is worth emphasizing that the decisive factor within C^{trans} is the distance, denoted as \mathbb{D}_{e_j} . Consequently, we can perceive this problem as a process of striking a balance between minimizing \mathbb{D}_{e_j} and server acquisition cost. The cost for e_j in stage I can be defined as: $\mathbb{C}_{e_j} = C_{e_j}^{alloc} + \mathbb{D}_{e_j}$. The objective function of stage I is defined as follows:

$$\begin{aligned} \mathbf{P}_2 : \text{minimize } & \sum_{e_j \in E} \mathbb{C}_{e_j}(\Delta), \\ \text{s.t. } & (4)\text{--}(6). \end{aligned} \quad (9)$$

\mathbf{P}_2 is the objective function of stage I. In stage I, we focus on the server deployment phase, aiming to achieve a balance in this COP problem.

Theorem 1: The problem of server deployment under distance thresholds, resource limitations, and connectivity requirements constraints to maximize profit for multiple users is an NP-hard problem.

Proof: The proof is given in Appendix. ■

B. Stage II

In stage II, we will address the remaining portion of \mathbf{P}_1 . In stage I, we have already considered the key factor \mathbb{D}_{e_j} for minimizing C^{trans} . However, to compute transmission cost, the service placement strategy of stage II is required. Considering that Stage I precedes Stage II, these two stages share the cost C^{trans} . Consequently, we have deferred the calculation of C^{trans} to stage II. Through stage I, \mathbb{D}_{e_j} has already become relatively small. In stage II, we just need to adjust the service placement

strategy to maximize $\mathbb{O}_{e_j} = I_{e_j} - (C^{trans} + C_{e_j}^{comp})$. Therefore, while these two stages are independent, they are interconnected. The objective function of stage II is defined as follows:

$$\begin{aligned} \mathbf{P}_3 : \text{maximize } & \sum_{e_j \in E} \mathbb{O}_{e_j}(S_{(e_j)}), \\ \text{s.t. } & (6)\text{--}(8). \end{aligned} \quad (10)$$

\mathbf{P}_3 is the objective function of stage II. After stage I, both E and Δ have been determined. In stage II, we only need to adjust the service placement strategy within each server to maximize profit while considering resource limitations.

Proposition 1: The approximate optimal solution obtained by sequentially solving \mathbf{P}_2 and \mathbf{P}_3 can be approximated as the near-optimal solution to \mathbf{P}_1 .

Proof: The proof is given in Appendix. ■

V. ANALYSIS AND SOLUTIONS

A. Stage I

The deployment of servers is framed as a COP within the MDP framework. At each time step, the agent is located at a specific base station and interacts with the environment, while also facing certain penalties.

Definition 1 (State Space): In our definition, let $Y(t) = [L_B, [a(1), a(2), \dots, a(t-1)], \{e_1 : wl_1, e_2 : wl_2, \dots\}]$ represents the state space at time step t which is divided into three parts: locations of all base stations, previously taken actions, and current servers' load status.

The state space defines all possible states of the problem. The states in the first part remain unchanged as the agent moves. We introduce the second and third parts of states that dynamically update as the agent moves to better comply with resource limitations.

Definition 2 (Action Space): In our definition, the joint action space for all base stations is denoted as $A = \{A_{b_1}, A_{b_2}, \dots, A_{b_k}, \dots, A_{b_{|B|}}\}$, where $A_{b_k} = B_{(b_k)} \times G$. For $a(t)$ at time step t , the first part is the chosen base station (denoted as $a^b(t)$), and the second part is the chosen server specification (denoted as $a^g(t)$).

The action space refers to the set of all possible actions that an agent can take. In our definition, the action space is divided into two parts, comprising the neighbor vectors for each base station and the server specifications to be determined. Therefore, the action space of each base station is defined as the Cartesian product of the set of potential action spaces (denoted as $B_{(b_k)}$) and the set of server specifications (denoted as G).

Definition 3 (Penalty Function): The penalty function can be defined as follows:

$$P(t) = d_{(y(t), a^b(t))} + (h(t) + \nu) \cdot z + Pr(t). \quad (11)$$

Here, $d_{(y(t), a^b(t))}$ is the distance between $y(t)$ and $a^b(t)$, and the initial value of z is set to 0. If $a^b(t)$ is not in the list of servers, the value of z is set to 1.

In order to reduce the transmission cost, the distance as the first factor must be added to the penalty function. But to prevent

agents from adding additional edge servers to the network while servers are available for connection, we introduce a constant ν into the penalty after adding a new edge server, i.e., $\nu \geq D_{th}$.

We need to assign a priority value, denoted as Pr , to each base station. This value indicates the priority of selecting the base station as the deployment location of the edge server. The priority of $a^b(t)$ is represented as follows:

$$Pr(t) = \frac{\mathbb{D}_{a^b(t)}}{wl_{a^b(t)} + |B_{(a^b(t))}|}. \quad (12)$$

Here, $\mathbb{D}_{a^b(t)}$ is the average distance between base stations in neighbor vectors of $a^b(t)$, and $wl_{a^b(t)}$ represents the workload of $a^b(t)$, and $|B_{(a^b(t))}|$ denotes the number of neighbor vectors of $a^b(t)$.

After the server deployment location is determined, the selection of server specifications must meet the actual load requirements to prevent resource waste. The reasonableness of $a^g(t)$ is defined as follows:

$$h(t) = |Z_{a^g(t)} - \mathbb{W}_{a^b(t)}|. \quad (13)$$

In the above formula, $Z_{a^g(t)}$ represents the storage capacity of the server that corresponds to the $a^g(t)$. $\mathbb{W}_{a^b(t)}$ represents the sum of the workload of base station in $B_{a^b(t)}$ that has not yet been connected to a server.

Definition 4 (Q-Table): We create a two-dimensional matrix of size $|B| \times (|B| \times G)$ as our Q-table, where $|B| \times G$ represents the size of the action space. We use penalty rather than reward for updating the Q-table, thus our Q-table update process is as follows:

$$Q(y(t), a(t)) \leftarrow Q(y(t), a(t)) + \alpha \times \left[P(t) + \gamma \min_{a \in A} Q(y(t+1), a) - Q(y(t), a(t)) \right]. \quad (14)$$

In the Q-table, states are represented as rows and actions as columns. Entries in the Q-table store the estimated penalty of taking a specific action in a specific state. In order to solve the problem of large state space, we only consider the locations of all base stations to define our Q-table. Forbidden Actions (FA) is a two-dimensional matrix with the same shape as the Q-table, but its entries are binary variables indicating whether the agent can take the specific action in that state.

The reason we chose Q-learning instead of more powerful deep reinforcement learning (DRL) methods like Proximal Policy Optimization (PPO) [39] or Soft Actor-Critic (SAC) is that neural networks may not be suitable for our modeling needs. The main reason lies in the differing number and location of neighbors around each base station, which results in inconsistent action space dimensions for each base station. In DRL, there are two possible solutions: First, designing a network for each base station, which would incur significant storage and inference time overhead; second, making decisions for all base stations simultaneously, which is also unsuitable because the total action space dimensions would grow exponentially with the number of base stations. This would lead to an excessive number of neurons in the neural network, reduced performance, and even

Algorithm 1: Server Deployment With Q (SDQ).

Input: $B, Y(t), A, FA$
Output: E, Δ

- 1 Initialize $Q(y, a)$;
- 2 **for** $episode \leftarrow 1$ **to** MAX **do**
- 3 **for** $\tau \leftarrow 1$ **to** $|B|$ **do**
- 4 $y(\tau) \leftarrow L_{b_\tau}$;
- 5 **if** $y(\tau)$ **in** E **then**
- 6 $a(\tau) \leftarrow b_\tau$;
- 7 **else**
- 8 $Nei \leftarrow$ Get actions based on FA ;
- 9 $PA \leftarrow$ Get actions based on $Y(t), Nei$;
- 10 **if** $|PA| > 0$ **then**
- 11 $Q_{PA} \leftarrow$ **_get_QValues**(PA);
- 12 Choose $a(\tau)$ for $y(\tau)$ using ϵ -greedy policy from Q_{PA} ;
- 13 **else**
- 14 $Q_{Nei} \leftarrow$ **_get_QValues**(Nei);
- 15 Choose $a(\tau)$ for $y(\tau)$ using ϵ -greedy policy from Q_{Nei} ;
- 16 Take action $a(\tau)$, $y(\tau+1) \leftarrow L_{b_{\tau+1}}$, and obtain P ;
- 17 Update $E, Y(t), \Delta, FA, Q(y, a)$;
- 18 **Reset**()
- 19 **return** E, Δ

server memory overload. Therefore, we designed a Q-table that includes all action spaces and uses FA to filter out actions outside the neighbor list of the corresponding base station, addressing the problem of large and variable action spaces.

Definition 5 (ϵ -greedy): In each iteration, the agent adopts an ϵ -greedy strategy to select actions for each state from the action space defined based on FA. The parameter ϵ represents the probability of selecting a random action, and $1 - \epsilon$ represents the probability of selecting an action based on the greedy strategy. It is determined as follows:

$$\epsilon = \frac{20}{T + 100}, \quad (15)$$

where T is the number of episodes. This formula ensures a high exploration rate at the beginning of training, allowing the agent to explore various actions. As T increases, ϵ gradually decreases, reducing the exploration rate and enabling the agent to shift to a greedy strategy, selecting actions that maximize the Q-values.

Algorithm 1 represents our proposed SDQ method. In line 3, the agent moves between base stations. In lines 5 to 6, if b_k where the agent is located has a deployed server, a connection is established with the server. Within lines 8 to 15, when b_k where the agent is located does not have a deployed server, we determine feasible actions based on FA, denoted as Nei . We also extracted the base stations where servers have already been deployed from Nei , denoted as PA . The Q-values of these actions are calculated. Then, we select $a(\tau)$ from these Q-values based on the ϵ -greedy policy. In lines 16 to 17, the action is taken,

and the relevant parameters are updated. In line 18, the **Reset()** refers to the process of resetting the E , $Y(t)$, Δ , and FA. Finally, E and Δ will be returned.

B. Stage II

1) *Problem Transformation*: For each type of service, the number of requests received by the server may not be less than the actual number of deployed replicas. Hence, (8) can be transformed as follows:

$$\sum_{s_w \in S(e_j)} M_{e_j}^{s_w} \leq \mu_{e_j}, \quad \forall e_j \in E. \quad (16)$$

The service placement problem within a single server can be transformed into a constrained INLP problem. The reduction to the standard form of the Interior-Point algorithm is as follows:

$$\begin{aligned} \mathbf{P}_4 : \text{minimize} \quad & f(x), \\ \text{s.t.} \quad & \zeta_i(x) \leq 0, \quad i = 1, 2. \end{aligned} \quad (17)$$

In the above formula, $x = S_{(e_j)}$, $f(x) = -\mathbb{O}_{e_j}(S_{(e_j)})$, $\zeta_1(x)$, $\zeta_2(x)$ are equivalent to (6) and (16). Equation (7) translates to the lower bound of x being zero.

We apply the interior point barrier method to transform the initial constrained problem into an unconstrained one. We utilize the function $\varphi(\eta) = 0(\eta \leq 0)$, otherwise $\varphi(\eta) = +\infty(\eta > 0)$ allowing us to reshape the objective function \mathbf{P}_4 into:

$$\mathbf{P}_5 : \text{minimize} \quad f(x) + \sum_{i=1}^2 \varphi(\zeta_i(x)). \quad (18)$$

When the solution of \mathbf{P}_5 satisfies the two constraints in \mathbf{P}_4 , $\varphi(\zeta_i(x)) = 0$, which implies that searching for the solution of \mathbf{P}_5 is equivalent to seeking a solution for \mathbf{P}_4 . If the solution of \mathbf{P}_5 doesn't satisfy any of the constraints in \mathbf{P}_4 , then \mathbf{P}_5 becomes infinite, making this solution unfeasible.

The function $\varphi(\eta)$ is non-differentiable. Therefore, we propose an approximation for $\varphi(\eta)$ using the function $\hat{\varphi}(\eta) = -\frac{1}{\psi} \cdot \log(-\eta)$. Here, we define $\Phi(x) = -\sum_{i=1}^2 \log(-\zeta_i(x))$, which is the log barrier function. Equation (18) is equivalent to:

$$\mathbf{P}_6 : \text{minimize} \quad \Phi(x) + \psi \cdot f(x). \quad (19)$$

As ψ increases, the accuracy improves and $\hat{\varphi}(\eta)$ approximates $\varphi(\eta)$, making the solution of \mathbf{P}_6 closer to the solution of \mathbf{P}_5 .

Theorem 2: The result of \mathbf{P}_6 gradually approximates the value of \mathbf{P}_4 when ψ increases to infinity, i.e., $\psi \rightarrow \infty$.

Proof: The proof is given in Appendix. ■

2) Service Placement With Interior Barrier Method (SPIB):

For \mathbf{P}_6 , which is an unconstrained problem, can be solved using Newton's method. At the current value of ψ , the the k -th iteration point is defined as x_k , and the optimal solution is denoted as x_ψ^* . We define $\theta(x) = \Phi(x) + \psi \cdot f(x)$. Newton's method involves approximating the objective function with a quadratic function at the iteration point x_k , and approximating the minimum point of the quadratic function as the minimum point of the objective function. The second-order Taylor expansion of $\theta(x)$ at x_k is given by $\theta(x_k) + \nabla^\top \theta(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^\top \nabla^2 \theta(x_k)(x - x_k) + o(\|x - x_k\|)^2$. We neglect

Algorithm 2: Service Placement With Interior Barrier Method (SPIB).

Input: $Z_{e_j}, \mu_{e_j}, \mathbb{D}_{e_j}, N_{(e_j)}^{s_w} |_{s_w \in S_{e_j}}, i_{s_w} |_{s_w \in S_{e_j}}$
Output: $S_{(e_j)}$

- 1 Initialize $x_\varphi^* = x_0, \psi = \psi^{(0)}, \omega, \varepsilon$;
- 2 **for** $episode \leftarrow 1$ **to** MAX **do**
- 3 **for** $episode \leftarrow 1$ **to** MAX **do**
- 4 **if** $\|\nabla \theta(x_\psi^*)\| < \varepsilon$ **then**
- 5 **Break**;
- 6 Calculate $d_{x_\varphi^*}^N, \Delta x_\varphi^*$;
- 7 $x_\varphi^* = x_\varphi^* + d_{x_\varphi^*}^N \cdot \Delta x_\varphi^*$
- 8 Update $S_{(e_j)} \leftarrow x_\psi^*$;
- 9 $\psi \leftarrow \omega \psi$;
- 10 **if** $\frac{2}{\psi} < \varepsilon$ **then**
- 11 **return** $S_{(e_j)}$;
- 12 **return** $S_{(e_j)}$

the infinitesimal term to obtain the approximation. Since $\theta(x)$ is a quadratic function, at the extremum point, $\nabla \theta(x_\psi^*) = 0$. Differentiating $\theta(x)$, we get $\nabla \theta(x) = \nabla \theta(x_k) + \nabla^2 \theta(x_k)(x - x_k) = 0$. We can then deduce $x = x_k - (\nabla^2 \theta(x_k))^{-1} \cdot \nabla \theta(x_k)$, setting $x = x_{k+1}$, and we obtain:

$$d_k^N = -(\nabla^2 \theta(x_k))^{-1} \cdot \nabla \theta(x_k), \quad (20)$$

d_k^N is the Newton direction for the k -th iteration.

For Newton's method, once the iteration direction is determined, the iteration step is assumed to be 1 by default, but this direction is not necessarily the direction of function value decrease. We can check this by taking the dot product of the current iteration direction and the gradient. If the dot product is negative, it indicates that the iteration direction is the descending direction, i.e., $-\nabla^\top \theta(x_k)(\nabla^2 \theta(x_k)) \nabla \theta(x_k)$. This condition holds only when the Hessian matrix $\nabla^2 \theta(x_k)$ is positive definite, which is a difficult condition to satisfy. Therefore, we introduce the concept of iteration steps. We use the method of linear search, as follows:

$$\Delta x_k = \arg \min_{\Delta x_k} \theta(x_k + \Delta x_k \cdot d_k^N), \Delta x_k \in \mathbb{R}, \quad (21)$$

When solving, we directly set $\frac{d\theta(x_k + \Delta x_k \cdot d_k^N)}{d\Delta x_k} = 0$ to solve for Δx_k . The iteration step can ensure that the iteration direction is the descending direction. Through successive iterations, the optimal solution is gradually approximated. When $\|\nabla \theta(x_k)\| < \varepsilon$, it indicates that the optimal point has been reached at the current ψ value.

For a single server, we use our proposed SPIB algorithm to determine the service placement strategy, as shown in Algorithm 2. In line 1, a set of various parameters is endowed with initial values. Among them, we define the initial iteration point as x_0 and the initial ψ . Lines 4 to 7 implement Newton's method to compute the optimal solution x_ψ^* for the current value of ψ . In line 9, the iteration index ψ is updated. Convergence

Algorithm 3: Tree-Based Branch-and-Bound (TDB).

```

Input:  $S_{(e_j)}, S$ 
Output:  $S_{(e_j)}$ 
1 Initialize  $sp \leftarrow \lfloor (S_{(e_j)}) \rfloor$ ,  $bound \leftarrow \mathbb{O}_{e_j}(sp)$ ,
    $root\_node, queue$ 
2  $queue.\_push(root\_node)$ ;
3 while  $queue$  not empty do
4    $node \leftarrow queue.\_pull()$ ;
5    $l\_node, r\_node \leftarrow \mathbf{Branch}()$ ;
6   Get the solutions for  $l\_node$  and  $r\_node$  using
   Algorithm 2;
7   if  $S_{(l\_node)}$  is feasible then
8     if  $\mathbb{O}_{e_j}(S_{(l\_node)}) > bound$  then
9       if  $S_{(l\_node)}$  is an integer solution then
10         $bound \leftarrow \mathbb{O}_{e_j}(S_{(l\_node)})$ ;
11         $sp \leftarrow S_{(l\_node)}$ ;
12      else
13         $queue.\_push(l\_node)$ ;
14   The same procedure is applied to the  $r\_node$ ;
15  $S_{(e_j)} \leftarrow sp$ ;
16 return  $S_{(e_j)}$ 

```

is established when the ratio $\frac{2}{\psi}$ falls below the predetermined threshold ε . Upon detecting convergence, the final service placement strategy vector $S_{(e_j)}$ is returned.

3) *Tree-Based Branch-and-Bound (TDB)*: Because the Algorithm 2 produces fractional solutions, we propose Algorithm 3 to find integer solutions that are approximate to the optimal solution. There are four main steps in Algorithm 3: initial bounding, branching, bounding, and pruning. In the initial bounding step, we directly round down the result of Algorithm 2 to obtain the initial integer solution. We substitute the initial integer solution into \mathbf{P}_3 to obtain the initial lower bound.

The branching step involves the selection of variables from the current leaf's solution that do not satisfy integer conditions. Subsequently, we generate two branches. The left and right child nodes will inherit constraints from the parent node. For the left child node, we introduce an additional constraint. The purpose of the constraint is to ensure that when solving for the left child node's solution, the values at the previously determined index position are less than or equal to the floor value of the value in the parent node's solution at the same index position. As for the right child node, the value should be greater than or equal to the ceiling value of the parent node's solution. Each child node can compute the objective function value under the current constraints to update the lower bound, thereby approximating the optimal integer solution. Finally, we combine bounding and pruning to effectively reduce the search space.

As depicted in Algorithm 3, line 1 initializes the initial integer solution and the lower bound. In line 2, we enqueue the root node. In lines 3 to 4, we commence processing the nodes in the queue. In lines 5 to 6, we perform the branching step and utilize Algorithm 2 to obtain the solutions and profit for the left and

right child nodes. In lines 7 to 14, we execute the bounding and pruning steps. Finally, we return the integer optimal solution in lines 15 to 16. By applying the SPIB-TDB algorithm to each server, we can obtain a global service placement strategy that maximizes profit.

C. Distance-and-Utilization Balance Algorithm (DUBA)

In this section, we consider how to determine the service deployment strategy when new users join. Maintaining the user, base station, and server connections established in stage I and II does not necessarily yield optimal performance. This is because, for new users, even if they fall within the coverage of multiple base stations or servers, their requests can still only be sent to a specific base station and server, i.e., (4). In fact, for users, if request forwarding is no longer fixed, it will lead to better overall network performance. Therefore, at this stage, for new users, we need to determine which base station and server they should connect to, maximizing user low-latency requirements and enhancing the operator's profit.

Users will usually preferentially connect to servers that are relatively close to each other. At the same time, since distance directly impacts the transmission cost of MNOs, we need to consider distance as a crucial consideration to maximize profit. When processing user requests on the server, there is a certain transmission cost incurred, which we define as connection distance. The connection distance (d_{u_v, e_j}) is defined as the sum of two distances:

- Distance between users and base stations (d_{u_v, b_k}).
- Distance between base stations and servers (d_{b_k, e_j}).

The connection distance is given by: $d_{u_v, e_j} = d_{u_v, b_k} + d_{b_k, e_j}$.

We model the entire network as a graph and employ Dijkstra's algorithm to calculate the shortest distance and path between every users and servers. Through this approach, we can determine which base station users should connect to when choosing to connect to a particular server. This optimization of network configuration aims to minimize the transmission cost of MNOs.

For users, the optimal choice does not only depend on the nearest server, but also needs to consider the utilization of the server. In some cases, a server that is not very far away but has low utilization may be better suited to handle service requests from a specific user, thereby providing greater profit to the operator. This is because sending task requests to a highly utilized server may result in forwarding to the cloud for processing, which in turn may cause higher transmission cost. Therefore, we include server utilization as one of the considerations.

The utilization of a server (ρ_{e_j}) is defined as the ratio of the actual workload (wl_{e_j}) to the maximum storage capacity (Z_{e_j}): $\rho_{e_j} = wl_{e_j}/Z_{e_j}$, where $wl_{e_j} = \sum_{s_w \in S_{(e_j)}} N_{e_j}^{s_w}$. Lower server utilization suggests that there is more available capacity on the server, increasing the likelihood that user service requests can be processed within the edge server without the need for forwarding to the cloud. Operators can achieve greater profit when server deployments achieve more balanced loads.

Definition 6 (Combination Weight): The combination weight (ι_{e_j}) is defined as a weighted combination of d_{u_v, e_j} and ρ_{e_j} ,

Algorithm 4: Distance-and-Utilization Balance Algorithm (DUBA).

Input: B, E, Δ, U
Output: $\Delta(U), \sum_{e_j \in E} S(e_j), \sum_{u_v \in U} \mathbb{O}_{u_v}$

- 1 Initialize $\mathbb{G}, edges, \iota_E$;
- 2 **for** $edge$ in $edges$ **do**
- 3 $wedge \leftarrow$ Get the length of $edge$;
 // weight of edge
- 4 **Add** $edge, wedge$ to \mathbb{G} ;
- 5 $paths, paths_length \leftarrow$ Execute Dijkstra's algorithm;
- 6 **for** u_v in U **do**
- 7 **for** e_j in E **do**
- 8 Calculate $d_{u_v, e_j}^{std}, \rho_{e_j}$;
- 9 $\iota_{e_j} \leftarrow v \cdot d_{u_v, e_j}^{std} + \beta \cdot \rho_{e_j}$;
- 10 **Add** ι_{e_j} to ι_E ;
- 11 Choose e_j with the lowest ι_{e_j} from ι_E ;
- 12 Update $\Delta(U)$ based on $u_v, paths_length[1], e_j$;
- 13 Update $S(e_j)$ based on $S(u_v)$;
- 14 $\mathbb{O}_{u_v} \leftarrow \mathbb{O}_{e_j}(S(u_v))$;
- 15 **return** $\Delta(U), \sum_{e_j \in E} S(e_j), \sum_{u_v \in U} \mathbb{O}_{u_v}$;

with weights represented by v and β respectively:

$$\iota_{e_j} = v \cdot d_{u_v, e_j} + \beta \cdot \rho_{e_j}. \quad (22)$$

We introduced the combination weight for each server. From the user's perspective, our goal is to select those servers with a lower value. Given that distance and utilization are on different magnitudes, we normalized distance, mapping it to a range from 0 to 1:

$$d_{u_v, e_j}^{std} = \frac{(d_{u_v, e_j} - D_{\min})}{D_{\max} - D_{\min}}. \quad (23)$$

By applying our proposed algorithm to each user, we are able to determine the connection relationships between all users, base stations, and servers. Once the connection relationship is determined, we can adjust the service placement strategy based on the type of service request sent by the user to enhance the profit of MNOs. The profit here is \mathbf{P}_3 which is the objective function of stage II. When calculating transmission cost, the average distance (\mathbb{D}_{e_j}) is no longer used, but the connection distance (d_{u_v, e_j}) is used. This is because we think from the user's perspective.

As shown in Algorithm 4, in line 1, we build the initial graph \mathbb{G} and its edges, and initialize the set ι_E . Moving to lines 2 to 4, we assign a weight to each edge and add them to the graph \mathbb{G} . Line 5 mark the beginning of Dijkstra's algorithm for calculating the shortest path and its length. For each element u_v in the set U , and each edge e_j in E , we calculate the weighted sum of distance and utilization to calculate ι_{e_j} (lines 6 to 10). The subsequent steps involve choosing the edge with the lowest ι_{e_j} from ι_E (line 11), updating $\Delta(U)$ based on the chosen edge (line 12). In line 13, we update the service placement strategy of e_j based on the user's service request type ($S(u_v)$). In line 14, we calculate the profit from serving u_v . The

TABLE II
PARAMETERS AND VALUES

Parameter	Meaning	Values
α	Learning Rate	0.4, 0.5, 0.6 , 0.7, 0.8
γ	Discount Factor	0.5, 0.6, 0.7 , 0.8
ω	Barrier Parameter	20, 50 , 220
ε	Convergence Threshold	10^{-5} , 10^{-6} , 10^{-7}
v	Distance Weight	0.2, 0.3, 0.4 , 0.5, 0.6, 0.7, 0.8
β	Utilization Weight	0.8, 0.7, 0.6 , 0.5, 0.4, 0.3, 0.2

algorithm concludes by returning the results: $\Delta(U)$, the sum of $S(e_j)$ for all e_j in E , and the sum of \mathbb{O}_{u_v} for all u_v in U .

D. Complexity Analysis

Please note that in Algorithm 1, the agent iterates through each base station and updates the Q-table, resulting in a complexity of $O(|B|^2)$. In Algorithm 2, the calculation of the convergence steps k is defined by equation: $2/\psi = 2/(\omega^k \cdot \psi^{(0)}) < \varepsilon$. Consequently, the complexity can be expressed as $k > \log(\frac{2}{\varepsilon \cdot \psi^{(0)}}) / \log \omega$. In Algorithm 3, due to its tree-based nature, where the depth of the tree is n , the complexity of Algorithm 3 is $O(2^n)$. In Algorithm 4, time complexity of executing Dijkstra's algorithm: $O((|V| + |\mathbb{E}|) \cdot \log |V| \cdot |U| \cdot |E|)$, where $|V| = |U| + |B| + |E|$, $|\mathbb{E}| = |U \cdot B \cdot E|$. Thus, the total complexity of our algorithm is $O(|B|^2 + \log(\frac{2}{\varepsilon \cdot \psi^{(0)}}) / \log \omega + 2^n + (|V| + |\mathbb{E}|) \cdot \log |V| \cdot |U| \cdot |E|)$. The simplified complexity is $O(2^n)$.

VI. EXPERIMENTAL EVALUATION

A. Basic Setting

To evaluate the performance of our proposed SDQ, SPIB, and TDB algorithms, we employed a dataset obtained from China Telecom Shanghai, which encompasses relevant information about 3,233 base stations and connected users over a span of 30 days in June 2014 [12], [13], [26]. For clarity in the subsequent experiments, the SPIB and TDB algorithms will be collectively referred to as the SPIB-TDB algorithm. Additionally, to demonstrate the performance of our proposed algorithms across different scales, we varied the number of base stations to 10, 20, 50, and 100, with corresponding user numbers set to 100, 300, 500, and 1000, respectively. Heterogeneous servers were configured with storage capacities of $\{30, 70, 150\}$, processing rates of $\{20, 60, 140\}$, a distance threshold of 9km, and cost of $\{10000, 20000, 40000\}$. The service benefit ratio coefficient was set to 1.3, and the ratio coefficient between computational and transmission cost was set to 1.5. Our experimental setup employed a desktop computer equipped with a 13th Gen Intel(R) Core(TM) i7-13700KF CPU and 32GB of memory. The parameters used in our proposed algorithm are summarized in Table II.

Referring to the experimental evaluation in [2], [10], and [13], [14], [15], [16], we introduced five baselines in comparison with our proposed SDQ algorithm:

- *Top-K*: The preference is to select base stations with the highest workload.

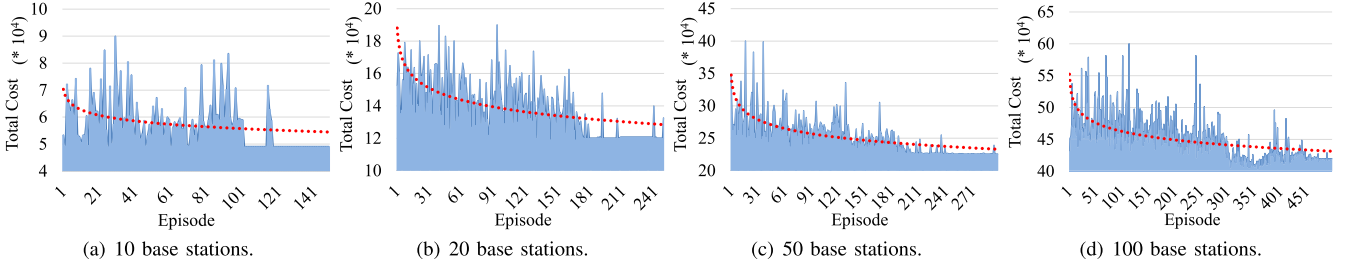


Fig. 2. Performance of SDQ by considering different numbers of base stations.

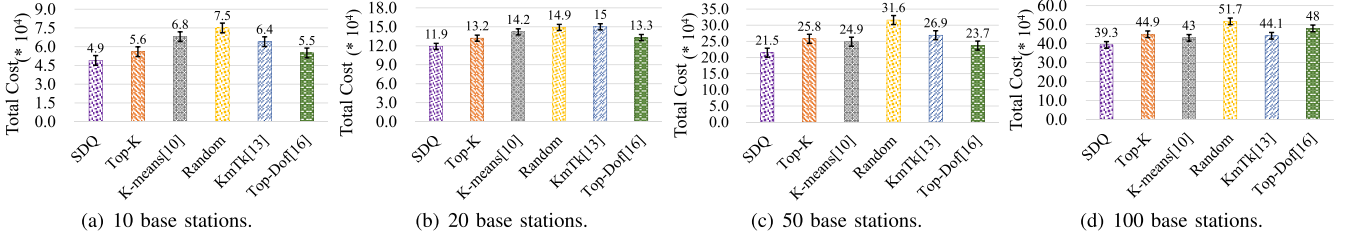


Fig. 3. Cost comparison for different numbers of base stations.

- *K-means*: This algorithm is employed iteratively, adjusting the parameter K to meet all constraints, which is modified from [10].
- *Top-DoF*: The preference is to select base stations with a higher number of neighbors [16].
- *K-means and Top-K (KmTK)*: The K-means algorithm is first executed, followed by the Top-K algorithm. More details can be seen from [13].
- *Random*: We will randomly select the locations for deploying the edge servers.

In stage II, we introduced three baselines in comparison with our proposed SPIB-TDB algorithm:

- *Greedy-popular (GP)*: Priority is given to placing services with the highest request rates.
- *Greedy-popular and income (GPI)*: Services with high request rates and considerable income are prioritized.
- *Random*: This algorithm involves determining placement strategy for services in a completely random manner.

For newly arrived users' service placement strategy, we introduce three baselines to compare with our proposed DUBA:

- *Distance-only-Algorithm (DOA)*: Each user's service request is forwarded to the nearest base station and server.
- *Utilization-only-Algorithm (UOA)*: Within the range of the transmission distance threshold, each user's service request is forwarded first to the least utilized server.
- *SDQ*: Each user's service request is forwarded according to the connection relationship in stage I.

B. Experiment Results

1) *Stage I*: In Algorithm 1, through continuous iterations, the agent gradually approximates the optimal actions, ultimately converging to a stable state (see Fig. 2). We tested the algorithm with varying numbers of base stations, 10, 20, 50, and 100, with

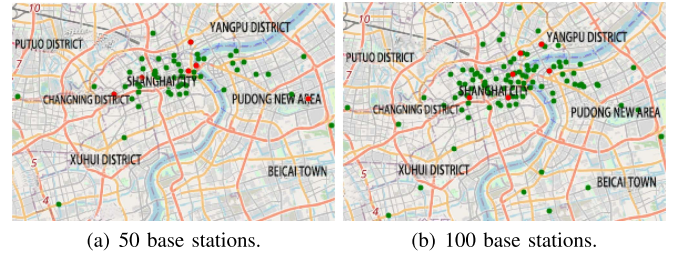


Fig. 4. Distribution of base stations and servers.

corresponding user counts set to 100, 300, 500, and 1000, respectively. The vertical axis in the graph represents the total cost, which corresponds to the objective function value of stage I (i.e., P_2). Lower values indicate superior algorithm performance. The horizontal axis represents the number of iterations. The fluctuation in the Total Cost value reflects the learning process of the agent through multiple attempts. As the number of base stations increases, the action space expands, requiring more iterations to discover the best actions. In Fig. 4, the distribution of base stations and servers is illustrated under different numbers of base stations using the SDQ algorithm. Fig. 9 displays the inference time of the SDQ algorithm under different numbers of base stations. The results indicate that the inference time increases with the number of base stations.

Fig. 3 displays the total cost for all algorithms across different numbers of base stations, with corresponding user counts set to 100, 300, 500, and 1000, respectively. As the number of base stations increases, we observe that the SDQ algorithm always maintains a relatively low total cost. Other algorithms have higher total cost. Notably, the K-means method performs well in minimizing transmission cost. However, its tendency to deploy servers to each base station results in higher server

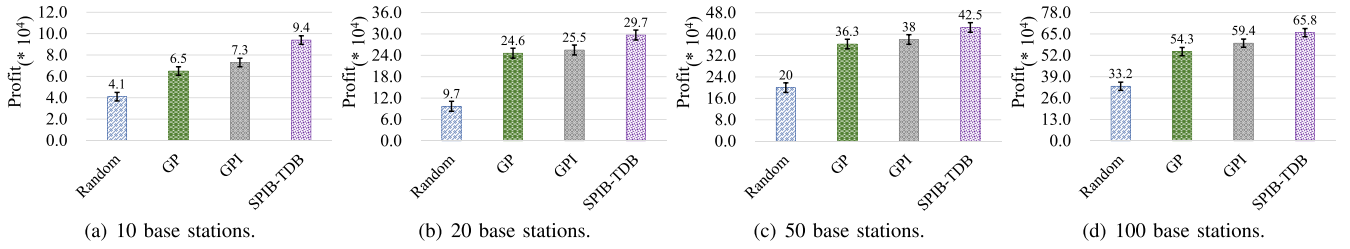


Fig. 5. Profit comparison for stage II (stage I : SDQ) under different numbers of base stations.

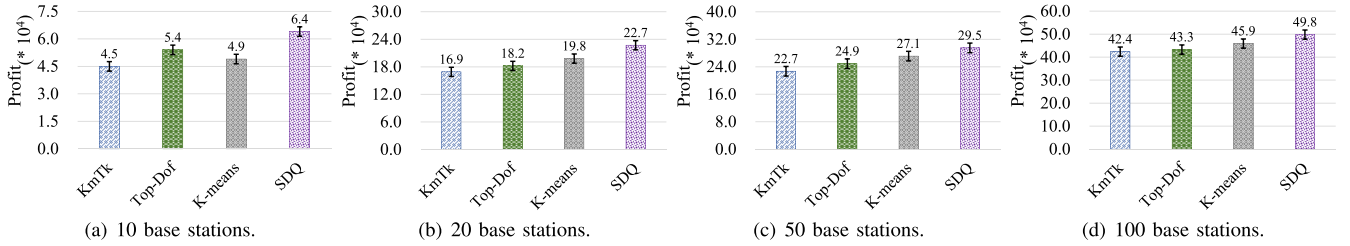


Fig. 6. Profit comparison for stage I (stage II : SPIB-TDB) under different numbers of base stations.

acquisition cost. As a result, its total cost is higher than our proposed algorithm. This trend is also present in other algorithms that utilize clustering. In the Top-DoF method, prioritizing nodes with the highest number of neighbors might lead to a concentration of servers in certain areas, resulting in uneven load distribution. The Top-K method, which selects servers based on their workload ranking, might not adequately consider the overall network topology and connectivity. From these experiments, we can conclude that SDQ exhibits the lowest total cost, as it simultaneously considers two objectives: 1) transmission cost and 2) server acquisition cost.

2) *Stage II*: since stage II builds upon the result of stage I, the experiments in stage II maintain a consistent utilization of our proposed SDQ algorithm for stage I. Distinct algorithms are employed for stage II to compare the performance, as illustrated in Fig. 5. The vertical axis Profit represents the objective function value of our stage II (i.e. P_3), where a higher value indicates better algorithm performance. It can be observed from Fig. 5 that SPIB-TDB algorithm that we proposed yields the highest Profit. Comparing the SPIB-TDB algorithm with the GP algorithm highlights the inadequacy of focusing only on the popularity of services. However, although the GPI algorithm takes into account both the popularity of services and the income they bring, it fails to consider the associated higher computation and storage cost of these services. Therefore, its algorithm performance is still lower than our SPIB-TDB algorithm.

In Fig. 6, the focus is on our overall objective function P_1 . This experimental design provides a more comprehensive illustration of the excellent performance achieved when combining our proposed SDQ with SPIB-TDB methods. In stage II, we consistently employ our proposed SPIB-TDB algorithm. However, in stage I, we utilize three baseline methods that demonstrated relatively good performance in previous experiments: KmTk,

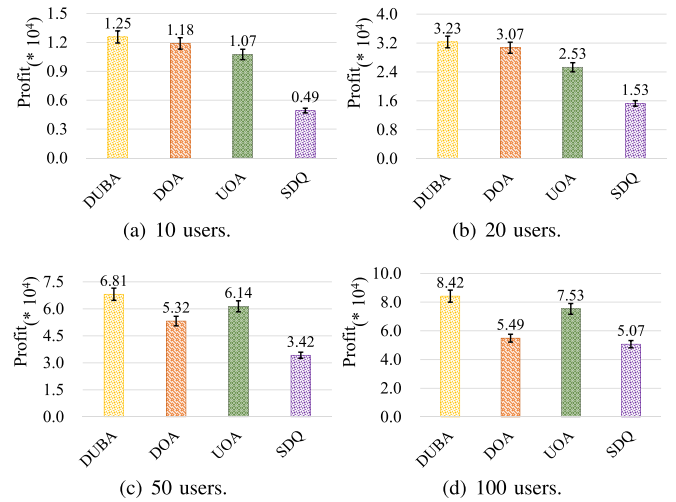
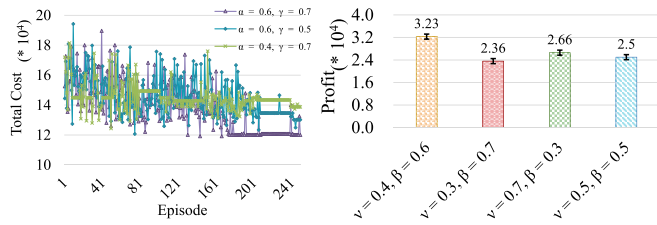


Fig. 7. Profit comparison for different algorithms for new users.

Top-dof, and K-means. In Fig. 6, the combination of SDQ and SPIB-TDB algorithms still yields the highest profit. Other algorithms perform poorly in stage I, as indicated in Fig. 3. Consequently, they yield lower overall profit, even when employing our proposed SPIB-TDB algorithm in stage II.

3) *DUBA*: In Fig. 7, we provide a detailed comparison of the performance of four different algorithms. Observing the gradual increase in the number of users from 10 to 100, we can see that the DUBA algorithm always shows the best profit level. In comparison, the profit levels of DOA and UOA algorithms are second, while the SDQ algorithm shows the lowest profit level. When the number of users is low, the likelihood of server overload is low, making it better for users to offload tasks to



(a) Convergence of SDQ under different learning rate and discount under Different v and β Weights. factor.

Fig. 8. Performance Comparison with Different Parameters under 20 base stations.

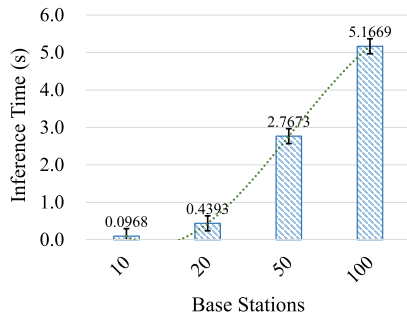


Fig. 9. Inference time of SDQ under different numbers of base stations.

the nearest server, i.e., DOA. However, as the number of users increases, servers cannot handle all the requests from nearby users. Continuing to use the DOA algorithm results in many requests being forwarded to the cloud, increasing transmission delays. Considering server utilization for scheduling, i.e., UOA, yields better performance in this situation.

DUBA performs best in terms of performance because it comprehensively considers multiple factors such as distance and utilization, thereby more comprehensively meeting user needs. In comparison, DOA and UOA algorithms each focus on a single factor, resulting in inferior performance to DUBA in some cases. This highlights the importance and advantages of considering multiple factors in the service placement problem. In addition, we conducted an in-depth study of the SDQ algorithm and found that its relatively low profit level is mainly attributed to its fixed forwarding mechanism for service requests. Specifically, when a new user joins, the SDQ algorithm limits the user to only send service requests to a specific base station and server, resulting in uneven distribution of server offloading.

In Fig. 8, we present the performance comparison under different parameters. Fig. 8(a) shows the convergence behavior of SDQ under different learning rate and discount factor with 20 base stations and 300 users. The results indicate that when the learning rate $\alpha = 0.4$, the convergence is slower, and the cost is higher. When the discount factor $\gamma = 0.5$, the algorithm emphasizes immediate rewards, but the cost is still higher compared to our parameter settings. This suggests that careful tuning of the learning rate and discount factor is necessary to balance the convergence speed and the final cost. Fig. 8(b) compares the performance of DUBA under different v and β weights. When $v = 0.4$ and $\beta = 0.6$, the system achieves the highest

profit, while other weight combinations result in relatively lower profit. Furthermore, we need to adjust the weights of distance and utilization based on the actual situation to achieve system optimization.

VII. CONCLUSION

This paper explores server deployment and service placement in multi-user scenarios to boost MNOs' profit, considering constraints such as distance, resources, and connectivity. We prove the problem is NP-hard and use a two-stage method. Our SDQ and SPIB-TDB algorithms leverage time sequences with varying validity, while DUBA optimizes new user placements based on distance and utilization. Experiments with Shanghai Telecom's dataset validate our algorithms.

Future work will extend this research to the dual time scale of service placement and task offloading, aiming to improve performance and service quality by optimizing the update frequency. In addition, the use of more powerful DRL algorithms will also be considered.

REFERENCES

- [1] Z. He, K. Li, and K. Li, "Cost-efficient server configuration and placement for mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 9, pp. 2198–2212, Sep. 2022.
- [2] A. Mazloomi, H. Sami, J. Bentahar, H. Otrok, and A. Mourad, "Reinforcement learning framework for server placement and workload allocation in multiaccess edge computing," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1376–1390, Jan. 2023.
- [3] Y. Chen, Y. Sun, H. Yu, and T. Taleb, "Joint task and computing resource allocation in distributed edge computing systems via multi-agent deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 4, pp. 3479–3494, Jul./Aug. 2024.
- [4] R. Men, X. Fan, K.-L. A. Yau, A. Shan, and G. Yuan, "Hierarchical aerial computing for task offloading and resource allocation in 6G-Enabled vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 4, pp. 3891–3904, Jul./Aug. 2024.
- [5] Y. Zhou, Y. Zhang, H. Liu, N. Xiong, and A. V. Vasilakos, "A bare-metal and asymmetric partitioning approach to client virtualization," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 40–53, Jan.–Mar. 2014.
- [6] X. Jiao et al., "Deep reinforcement learning for time-energy tradeoff online offloading in MEC-enabled industrial Internet of Things," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3465–3479, Nov./Dec. 2023.
- [7] L. Wang and G. Zhang, "Joint service caching, resource allocation and computation offloading in three-tier cooperative mobile edge computing system," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3343–3353, Nov./Dec. 2023.
- [8] C. Wang et al., "Heterogeneous edge caching based on actor-critic learning with attention mechanism aiding," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3409–3420, Nov./Dec. 2023.
- [9] A.-V. Anttiroiko, P. Valkama, and S. J. Bailey, "Smart cities in the new service economy: Building platforms for smart services," *AI Soc.*, vol. 29, pp. 323–334, 2014.
- [10] X. Zhang, Z. Li, C. Lai, and J. Zhang, "Joint edge server placement and service placement in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11261–11274, Jul. 2022.
- [11] Z. Ning et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2628–2644, May 2023.
- [12] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 55–67, Jan. 2022.
- [13] Y. Guo et al., "User allocation-aware edge cloud placement in mobile edge computing," *Softw.: Pract. Experience*, vol. 50, no. 5, pp. 489–502, 2020.
- [14] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct.–Dec. 2017.
- [15] S. Wang et al., "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, 2019.

- [16] T. M. Ho and K.-K. Nguyen, "Joint server selection, cooperative offloading and handover in multi-access edge computing wireless network: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2421–2435, Jul. 2022.
- [17] S. K. Kasi et al., "Heuristic edge server placement in industrial Internet of Things and cellular networks," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10308–10317, Jul. 2021.
- [18] X. Xu et al., "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2910–2918, Apr. 2021.
- [19] C. Ling et al., "An edge server placement algorithm based on graph convolution network," *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 5224–5239, Apr. 2023.
- [20] Y. Qu et al., "Server placement for edge computing: A robust submodular maximization approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3634–3649, Jun. 2023.
- [21] Y. Xu, V. Chau, C. Wu, Y. Zhang, and Y. Zou, "Online joint placement and allocation of virtual network functions with heterogeneous servers," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8049–8058, Sep. 2020.
- [22] P. Han, Y. Liu, X. Zhang, and L. Guo, "Energy-efficient service placement based on equivalent bandwidth in cell zooming enabled mobile edge cloud networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12275–12290, Nov. 2022.
- [23] H. Sami, A. Mourad, H. Otrouk, and J. Bentahar, "Demand-driven deep reinforcement learning for scalable fog and service placement," *IEEE Trans. Serv. Comput.*, vol. 15, no. 5, pp. 2671–2684, Sep./Oct. 2022.
- [24] Y. Li et al., "Cooperative service placement and scheduling in edge clouds: A deadline-driven approach," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3519–3535, Oct. 2022.
- [25] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 3836–3851, Nov. 2022.
- [26] X. Shen et al., "Deep variational matrix factorization with knowledge embedding for recommendation system," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 1906–1918, May 2021.
- [27] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1459–1467.
- [28] J. Yang et al., "A fingerprint recognition scheme based on assembling invariant moments for cloud computing communications," *IEEE Syst. J.*, vol. 5, no. 4, pp. 574–583, Dec. 2011.
- [29] J. Fang and A. Ma, "IoT application modules placement and dynamic task processing in edge-cloud computing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12771–12781, Aug. 2021.
- [30] J. Li, B. Gu, Z. Qin, and Y. Han, "Graph tasks offloading and resource allocation in multi-access edge computing: A DRL-and-optimization-aided approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3707–3718, Nov./Dec. 2023.
- [31] Z. Teng, J. Fang, H. Yang, L. Yu, H. Chen, and W. Xiang, "Attention mechanism-aided deep reinforcement learning for dynamic edge caching," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10197–10213, Mar. 2024.
- [32] G. Qu et al., "TrimCaching: Parameter-sharing AI model caching in wireless edge networks," 2024, *arXiv:2405.03990*.
- [33] J. Wu, L. Wang, Q. Pei, X. Cui, F. Liu, and T. Yang, "HiTDL: High-throughput deep learning inference at the hybrid mobile edge," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4499–4514, Dec. 2022.
- [34] S. Chen, L. Wang, and F. Liu, "Optimal admission control mechanism design for time-sensitive services in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1169–1178.
- [35] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *Proc. 2020-IEEE Conf. Comput. Commun.*, 2020, pp. 267–276.
- [36] M. Li, Q. Zhang, and F. Liu, "Finedge: A dynamic cost-efficient edge resource management platform for NFV network," in *Proc. IEEE/ACM 28th Int. Symp. Qual. Serv.*, 2020, pp. 1–10.
- [37] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1357–1371, Jun. 2020.
- [38] L. Pan, L. Wang, S. Chen, and F. Liu, "Retention-aware container caching for serverless edge computing," in *Proc. 2022-IEEE Conf. Comput. Commun.*, 2022, pp. 1069–1078.
- [39] R. Zhang, K. Xiong, Y. Lu, P. Fan, D. W. K. Ng, and K. B. Letaief, "Energy efficiency maximization in RIS-assisted SWIPT networks with RSMA: A PPO-based approach," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 5, pp. 1413–1430, May 2023.



Juan Fang (Member, IEEE) received the M.S. degree from the Jilin University of Technology, Changchun, China, in 1997, and the Ph.D. degree from the College of Computer Science, Beijing University of Technology, Beijing, China, in 2005. In 1997, she joined the College of Computer Science, Beijing University of Technology. From 2015, she is the Professor of Beijing University of Technology. She currently works with the College of Computer Science, Beijing University of Technology.



Shen Wu (Student Member, IEEE) received B.Sc. degree in computer science and technology with Yan-shan University, Qinhuangdao, China. He is currently working toward the M.Sc. degree in electronic information with the College of Computer Science Technology, Beijing University of Technology, Beijing, China. He is also a Student Member of CCF.



Shuaibing Lu (Member, IEEE) received the Ph.D. degree in computer science and technology from Jilin University, Changchun, China, in 2019. She is currently a Lecturer with the College of Computer Science, Beijing University of Technology, Beijing, China. From 2016 to 2018, she is supported by the China Scholarship Council as a Visiting Scholar supervised by Prof. Jie Wu with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA.



Ziyi Teng (Student Member, IEEE) received the B.S. degree from the North China University of Water Resources and Electric Power, Zhengzhou, China, in 2019. She is currently working toward the Ph.D. degree with the Beijing University of Technology, Beijing, China. Her research focusses on mobile edge computing. She is a Student Member of CCF.



Huijie Chen (Member, IEEE) received the B.Eng degree from the School of Computer Science, Henan University of Economics and Law, Zhengzhou, China, in 2010, the M.Seng degree from the School of Computer Science, Taiyuan University of Science and Technology, Taiyuan, China, in 2013, and the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2020. He is currently with the School of Computer Science, Beijing University of Technology, Beijing. His research interests include smart sensing and mobile edge computing.



Neal N. Xiong (Senior Member, IEEE) received the Ph.D. degree in sensor system engineering from Wuhan University, Wuhan, China, in 2007, and in dependable communication networks from the Japan Advanced Institute of Science and Technology in 2008. Before joining Sul Ross State University, he was with Georgia State University, Northeastern State University, and Colorado Technical University (a Full Professor for about five years) for around 15 years. He is currently a Professor with the Department of Computer Science and Mathematics, Sul Ross State University, Alpine, TX, USA. His research interests include cloud computing, security and dependability, AI, data analysis, parallel and distributed computing, and networks.