

Profit-driven Optimization of Server Deployment and Service Placement in Multi-User Mobile Edge Computing

1st Juan Fang
Faculty of Information Technology
Beijing University of Technology
Beijing, China
fangjuan@bjut.edu.cn

2nd Shen Wu
Faculty of Information Technology
Beijing University of Technology
Beijing, China
wushen@emails.bjut.edu.cn

3rd Shuaibing Lu
Faculty of Information Technology
Beijing University of Technology
Beijing, China
lushuaibing@bjut.edu.cn

Abstract—Edge computing has emerged as a promising paradigm to fulfill the escalating demands of latency-sensitive and computationally intensive applications. In this context, efficient server deployment and service placement have become imperative to optimize performance and increase platform profit. In this paper, we investigate the problem of server deployment and service placement in a multi-user scenario, aiming to enhance the profit of Mobile Network Operators (MNOs) while considering constraints related to distance thresholds, resource limitations, and connectivity requirements. Then, we propose a novel two-stage method to decouple the problem, breaking down server deployment and service placement into two distinct yet interconnected stages. In stage I, the server deployment is formulated as a combinatorial optimization problem within the framework of a Markov Decision Process (MDP), where the state space, action space, and penalty function are defined to effectively model the issue. We propose the SDQ algorithm to establish a relatively stable server deployment strategy. In stage II, the service placement is formulated as a constrained integer linear programming problem. We propose the SPIB-TDB algorithm to optimize service placement. Extensive experimentation validates the exceptional performance of our proposed algorithms in enhancing the profit of MNOs.

Index Terms—mobile edge computing, profit-driven optimization, reinforcement learning, integer linear programming.

I. INTRODUCTION

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. 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 MNOs. 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.

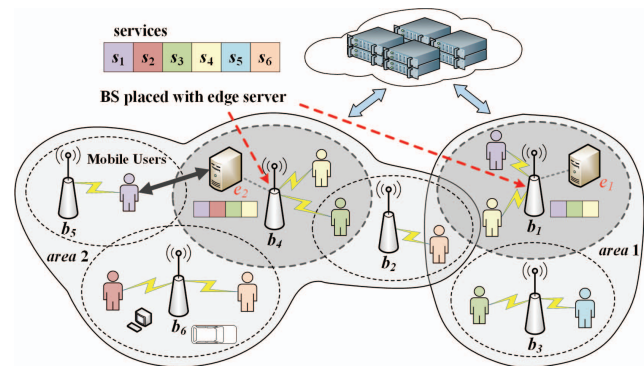


Fig. 1. An illustrating example.

A. Motivation and Challenges

Server deployment and service placement significantly impact the profit of MNOs. We turn to the example provided in Figure 1. Within this illustration, six base stations are depicted, with servers positioned at b_1 and b_4 . Some services have already been placed on edge servers, while the remaining services are all placed in the cloud. Each service replica serves a single user and generates income. In Figure 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. The service placement phase cannot only consider the service request rate. This will give rise to some issues and challenges: (i). How to determine the number, specifications of servers, and locations of servers capable of serving as many users as possible in different areas, given the cost of MNOs. (ii). How to determine service placement strategies within limited server capacity to increase the profit of MNOs.

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, breaking down server deployment and service placement into two distinct yet interconnected stages.
- 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 Integer Linear Programming (ILP) problem. Then, we propose a strategy for achieving the integer optimal solution with remarkable performance.
- We conducted extensive experiments to compare our strategy with several baselines based on a real base station dataset provided by Shanghai Telecom. Extensive experiments demonstrate the superior performance of the proposed algorithm.

The remainder of this paper is organized as follows. In Section II, we review related research. In Section III, we modeled the problem. In Section IV, we propose the problem of server deployment and service placement. In Section V, we propose the solution to the problem, while in Section VI, we illustrate the comparative experimental evaluation results. Finally, in Section VII, we conclude the paper.

II. RELATED WORK

A. Server Deployment

In their work, He et al. [1] developed a two-stage algorithm based on Binary and GA to obtain optimal and suboptimal placement schemes for ES. Mazloomi et al. [2] modeled the practical problem using reinforcement learning to solve the challenge of minimizing network latency and the number of edge servers. Ning et al. [3] formulated models for dynamic multi-user computation offloading and edge server deployment problems using stochastic game theory. Li et al. [4] 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. [5] proposed an approximate method using k-means and mixed-integer quadratic programming to balance the workload among edge clouds. Jia et al. [6] studied cloud placement and assignment of mobile users to clouds. But these papers do not consider the heterogeneity of servers

B. Service Placement

In their work, Zhang et al. [7] constructed a joint model for edge server deployment and service placement to maximize

MNOs' profit. Gao et al. [8] designed a two-stage algorithm to study the joint optimization of MEC access network selection and service layout. Wang et al. [9] introduced an online algorithm based on reinforcement learning to learn the optimal coordination scheme. Gao et al. [10] proposed an iterative-based algorithm to address the problem of joint optimization of MEC access network selection and service placement. Brik et al. [11] introduced a taboo search metaheuristic algorithm aimed at balancing the computational load. Wu et al. [12] presented a weighted cost model to minimize the execution time and energy consumption. Badri et al. [13] modeled the energy-aware application placement problem as a multi-stage stochastic program. Many existing studies assume that edge servers are pre-deployed at base stations [14]–[19]. In such cases, the connection between edge server deployment and service placement is severed. In addition, the overall profit of the MEC platform is also a critical factor to consider [20]. Telecom operators may find it challenging to bear the cost of deploying edge servers for every base station.

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 MODEL

A. The MEC Environment

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 .

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 C_{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 .

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

1) *Server Acquisition Cost*: 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 edge servers dealing with service requests, and the other is the transmission cost generated by cloud servers dealing with service requests. In the transmission cost generated by edge servers dealing with service requests, the cost from users to base stations is negligible. Now, we calculate the cost of service requests arriving at the edge servers. 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 transmission cost between cloud servers and edge servers is high, so when relying on cloud servers for processing, additional transmission cost will be incurred. 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 $C_{(b_k, e_j)}^{trans} = C_{(e_j, R)}^{trans} + C_{(e_j, b_k)}^{trans}$.

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:

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

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.

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:

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

\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, taking into account constraints related to distance thresholds and connectivity requirements.

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:

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

\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.

V. ANALYSIS AND SOLUTIONS

A. Stage I

The deployment of servers is framed as a combinatorial optimization problem within the MDP framework.

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.

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 $a^g(t)$).

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 our algorithm, the penalty function is used instead of the reward function in standard Q-learning. 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.

Definition 5 (Forbidden Actions): 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.

Algorithm 1 represents our proposed SDQ method. Our ultimate goal is to gradually converge the values within the Q-table. 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

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        $N_{ei} \leftarrow$  Get actions based on  $FA$ ;  
9        $PA \leftarrow$  Get actions based on  $Y(t), N_{ei}$ ;  
10      if  $|PA| > 0$  then  
11         $Q_{PA} \leftarrow$  _get_QValues( $PA$ );  
12        Choose  $a(\tau)$  for  $y(\tau)$  using  $\epsilon$ -greedy  
        policy from  $Q_{PA}$ ;  
13      else  
14         $Q_{N_{ei}} \leftarrow$  _get_QValues( $N_{ei}$ );  
15        Choose  $a(\tau)$  for  $y(\tau)$  using  $\epsilon$ -greedy  
        policy from  $Q_{N_{ei}}$ ;  
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  $E, \Delta$ 
```

$a(\tau)$ 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 $E, Y(t), \Delta$, 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, Equation (8) can be transformed as follows:

$$\sum_{s_w \in S(e_j)} M_{e_j}^{s_w} \leq \mu_{e_j}, \quad \forall e_j \in E. \quad (14)$$

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

$$\begin{aligned} \mathbf{P}_4 : \text{minimize} \quad & f(x) \\ \text{s.t.} \quad & \zeta_i(x) \leq 0, \quad i = 1, 2. \end{aligned} \quad (15)$$

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 Equations (6) and (14). 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:

$$\mathbf{P}_5 : \text{minimize} \quad f(x) + \sum_{i=1}^2 \varphi(\zeta_i(x)). \quad (16)$$

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.

Definition 6 (Log Barrier Function): 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 (16) is equivalent to:

$$\mathbf{P}_6 : \text{minimize} \quad \Phi(x) + \psi \cdot f(x). \quad (17)$$

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 1: 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: We employ the Lagrangian algorithm to transform \mathbf{P}_4 into the Lagrangian function: $L(x, \lambda) = f(x) + \sum_{i=1}^2 \lambda_i \zeta_i(x)$. The Lagrangian dual function is defined as: $g(\lambda) = \inf_x L(x, \lambda)$. This function provides a lower bound estimation for \mathbf{P}_4 . We use p^* to represent the minimum value of \mathbf{P}_4 , thus we have $g(\lambda) \leq p^*$. The solution of \mathbf{P}_4 must satisfy the Karush-Kuhn-Tucker (KKT) condition. Therefore, we can derive the following inequalities:

$$\zeta_i(x) \leq 0, \quad i = 1, 2, \quad (18)$$

$$\lambda_i \geq 0, \quad i = 1, 2, \quad (19)$$

$$\nabla f(x) + \sum_{i=1}^2 \lambda_i \cdot \nabla \zeta_i(x) = 0, \quad (20)$$

$$\lambda_i \cdot \zeta_i(x) = 0, \quad i = 1, 2. \quad (21)$$

For \mathbf{P}_6 , the function $\Phi(x)$ possesses favorable properties. It is a monotonically differentiable function. First, we provide the first derivative of $\Phi(x)$:

$$\nabla \Phi(x) = \sum_{i=1}^2 -\frac{1}{\zeta_i(x)} \cdot \nabla \zeta_i(x). \quad (22)$$

When $\psi > 0$, we define x_ψ^* as the optimal solution for \mathbf{P}_6 at the current value of ψ . The derivative of \mathbf{P}_6 is given by $\psi \nabla f(x_\psi^*) + \nabla \Phi(x_\psi^*) = 0$. By substituting Equation (22), we can divide both sides by ψ to obtain

$$\nabla f(x_\psi^*) + \sum_{i=1}^2 -\frac{1}{\psi \cdot \zeta_i(x_\psi^*)} \cdot \nabla \zeta_i(x_\psi^*) = 0. \quad (23)$$

Let $\lambda_i^* = -\frac{1}{\psi \cdot \zeta_i(x_\psi^*)}$, and Equation (23) becomes $\nabla f(x_\psi^*) + \sum_{i=1}^2 \lambda_i^* \cdot \nabla \zeta_i(x_\psi^*) = 0$. It can be observed that the current

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 $S_{(e_j)} = x_0, \psi = \psi^{(0)}, \omega, \varepsilon, \rho;$
- 2 **for** $episode \leftarrow 1$ **to** MAX **do**
- 3 **for** $episode \leftarrow 1$ **to** MAX **do**
- 4 Calculate $x_{\psi}^*, \frac{1}{2}\delta^2(x_{\psi}^*)$ by using Equations (28) - (30);
- 5 **if** $\frac{1}{2}\delta^2(x_{\psi}^*) < \varepsilon$ **then**
- 6 **break**;
- 7 Update $S_{(e_j)} \leftarrow x_{\psi}^*;$
- 8 $\psi \leftarrow \omega\psi;$
- 9 **if** $\frac{2}{\psi} < \varepsilon$ **then**
- 10 **return** $S_{(e_j)};$
- 11 **return** $S_{(e_j)}$

expression Equation (26) is now similar in form to Equation (20), which represents the derivative of the Lagrangian function $L(x, \lambda)$. We will now discuss the KKT conditions for \mathbf{P}_6 , and it can be deduced:

$$\zeta_i(x_{\psi}^*) \leq 0, i = 1, 2 \quad (24)$$

$$\lambda_i^* \geq 0, i = 1, 2 \quad (25)$$

$$\nabla f(x_{\psi}^*) + \sum_{i=1}^2 \lambda_i^* \cdot \nabla \zeta_i(x_{\psi}^*) = 0 \quad (26)$$

$$\lambda_i^* \cdot \zeta_i(x_{\psi}^*) = -\frac{1}{\psi}, i = 1, 2 \quad (27)$$

We can observe that as ψ tends toward infinity, $-\frac{1}{\psi}$ approximates 0, and the KKT conditions become increasingly similar.

After that, we discuss the error between the result of \mathbf{P}_6 and the value of \mathbf{P}_4 : $|f(x_{\psi}^*) - p^*|$. For the Lagrangian dual function $g(\lambda^*) = \inf_x L(x, \lambda^*)$, at the optimal point, $\nabla_x L(x, \lambda^*) = 0$. We have derived $\nabla_x L(x_{\psi}^*, \lambda^*) = 0$ in Equation (26), so $g(\lambda^*) = f(x_{\psi}^*) + \sum_{i=1}^2 \lambda_i^* \cdot \zeta_i(x_{\psi}^*) = f(x_{\psi}^*) - \frac{2}{\psi} \leq p^*$. Since p^* is the optimal solution of \mathbf{P}_4 , we conclude that $f(x_{\psi}^*) \geq p^*$. Thus, we can deduce that $p^* \leq f(x_{\psi}^*) \leq p^* + \frac{2}{\psi}$. It is evident that as $\frac{2}{\psi}$ increases, $f(x_{\psi}^*)$ progressively approximates p^* . ■

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 initial iteration point is defined as $x_{\psi}^* = x_0$. We let $\theta(x) = \Phi(x) + \psi \cdot f(x)$, and the algorithm update is given by:

$$\Delta x_{\psi} = -(\nabla^2 \theta(x_{\psi}^*))^{-1} \cdot \nabla \theta(x_{\psi}^*), \quad (28)$$

$$x_{\psi}^* = x_{\psi}^* + \rho \cdot \Delta x_{\psi}, \quad (29)$$

$$\delta^2(x_{\psi}^*) = \nabla^T \theta(x_{\psi}^*) \cdot (\nabla^2 \theta(x_{\psi}^*))^{-1} \cdot \nabla \theta(x_{\psi}^*). \quad (30)$$

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** $S_{(e_j)}$

Where $\nabla \theta(x_{\psi}^*)$ is the vector of first derivative, and $\nabla^2 \theta(x_{\psi}^*)$ represents the Hessian matrix. Equation (28) indicates the iteration step size. Equation (29) represents the iterative approach to the solution. Through successive iterations, the optimal solution is gradually approximated. The error between x_{ψ}^* and the optimal solution at the current value of ψ can be expressed by $\frac{1}{2}\delta^2(x_{\psi}^*)$. Therefore, Equation (30) can be used as the stopping criterion for iteration.

We use $\frac{2}{\psi}$ as the stopping criterion for our proposed algorithm. For a single server, we use our proposed SPIB algorithm to determine the service deployment 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 6 implement Newton's method to compute the optimal solution x_{ψ}^* for the current value of ψ . In lines 7 to 10, the iteration index ψ is updated. Convergence 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

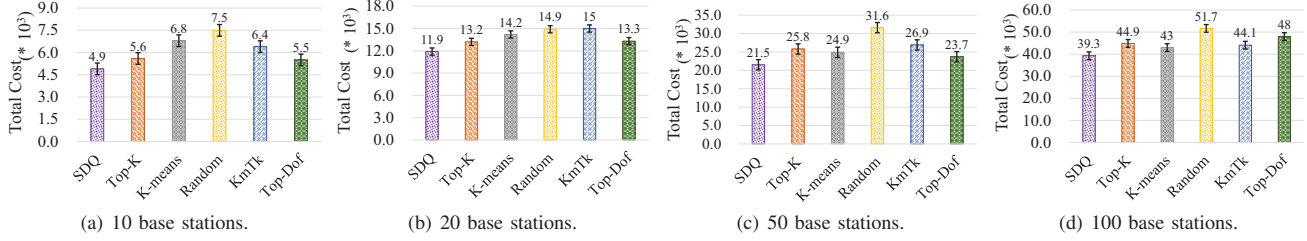


Fig. 2. Cost comparison for different numbers of base stations.

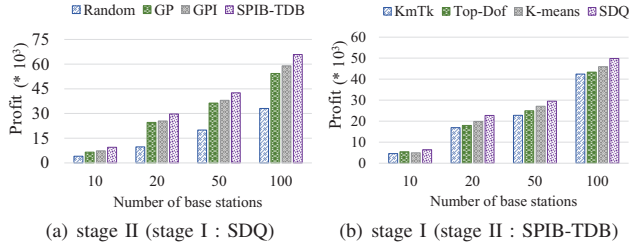


Fig. 3. Profit comparison for different stages.

current leaf's solution that do not satisfy integer conditions, and dividing them into two child nodes. 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. We set the initial constraints and the integer solution as the root node. 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.

C. 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 $k > \log\left(\frac{2}{\varepsilon \cdot \psi^{(0)}}\right) / \log \omega$. In Algorithm 3, due to its tree-based nature, where the depth of the tree is n , the complexity of Algorithm 3 is $O(2^n)$. Thus, the total complexity of our algorithm is $O(|B|^2 + \log\left(\frac{2}{\varepsilon \cdot \psi^{(0)}}\right) / \log \omega + 2^n)$, where $|B|$ and n represent the number of base stations and the number of service types, respectively.

VI. EXPERIMENTAL EVALUATION

A. Basic Setting

To evaluate the performance of our proposed SDQ and SPIB-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. The daily volume of service requests directed to each base station from users is indicative of the station's workload. Heterogeneous servers were configured with storage capacities of $\{30, 70, 150\}$, processing rates of $\{20, 60, 140\}$, a distance threshold of 9 km, and cost of $\{10000, 20000, 40000\}$. The service benefit ratio coefficient was set to 1.3, and the ratio coefficient between computation 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 32 GB of memory.

In stage I, we introduced five baselines in comparison with our proposed SDQ algorithm:

- **Top-K**: The preference is to select base stations with the highest workload.
- **K-means**: This algorithm is employed iteratively, adjusting the parameter K to meet thresholds, resource limitations, and connectivity requirements constraints.
- **Top-DoF**: The preference is to select base stations with a higher number of neighbors.
- **K-means and Top-K (KmTk)**: K-means is initially executed. Subsequently, the base station with the highest workload within each cluster is chosen as the edge server.
- **Random**: We will randomly select the locations for deploying the edge servers from among the base stations.

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)**: In the greedy algorithm, services with high request rates and considerable income are prioritized.
- **Random**: This algorithm involves determining placement strategy for services in a completely random manner.

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. Figure 2 displays the total cost for all algorithms across different numbers of base stations. 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 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 Figure 3 (a). 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 Figure 3 (a) that the 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 Figure 3 (b), 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, Top-dof, and K-means. In Figure 3 (b), the combination of SDQ and SPIB-TDB algorithms still yields the highest profit. Other algorithms perform poorly in stage I, as indicated in Figure 2. Consequently, they yield lower overall profit, even when employing our proposed SPIB-TDB algorithm in stage II.

VII. CONCLUSION

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. We employ a two-stage method to decouple the problem. On this basis, we proposed the SDQ and SPIB-TDB algorithms. Finally, we conducted experiments using a real base station dataset provided by Shanghai Telecom to verify the superior performance of our proposed algorithms.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (62202019,62276011,61202076), and

supported by the Beijing Municipal Natural Science Foundation (4192007), along with other government sponsors.

REFERENCES

- [1] He Z, Li K, Li K. Cost-efficient server configuration and placement for mobile edge computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2198-2212.
- [2] Mazloomi A, Sami H, Bentahar J, et al. Reinforcement Learning Framework for Server Placement and Workload Allocation in Multiaccess Edge Computing[J]. IEEE Internet of Things Journal, 2022, 10(2): 1376-1390.
- [3] Ning Z, Yang Y, Wang X, et al. Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing[J]. IEEE Transactions on Mobile Computing, 2021.
- [4] Li Y, Zhou A, Ma X, et al. Profit-aware edge server placement[J]. IEEE Internet of Things Journal, 2021, 9(1): 55-67.
- [5] Guo Y, Wang S, Zhou A, et al. User allocation-aware edge cloud placement in mobile edge computing[J]. Software: Practice and Experience, 2020, 50(5): 489-502.
- [6] Jia M, Cao J, Liang W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing, 2015, 5(4): 725-737.
- [7] Zhang X, Li Z, Lai C, et al. Joint edge server placement and service placement in mobile-edge computing[J]. IEEE Internet of Things Journal, 2021, 9(13): 11261-11274.
- [8] Gao B, Zhou Z, Liu F, et al. An online framework for joint network selection and service placement in mobile edge computing[J]. IEEE Transactions on Mobile Computing, 2021, 21(11): 3836-3851.
- [9] Wang S, Guo Y, Zhang N, et al. Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2019, 20(3): 939-951.
- [10] Gao B, Zhou Z, Liu F, et al. Winning at the starting line: Joint network selection and service placement for mobile edge computing[C]//IEEE INFOCOM 2019-IEEE conference on computer communications. IEEE, 2019: 1459-1467.
- [11] Brik B, Frangoudis P A, Ksentini A. Service-oriented MEC applications placement in a federated edge cloud architecture[C]//ICC 2020-2020 IEEE international conference on communications (ICC). IEEE, 2020: 1-6.
- [12] Goudarzi M, Wu H, Palaniswami M, et al. An application placement technique for concurrent IoT applications in edge and fog computing environments[J]. IEEE Transactions on Mobile Computing, 2020, 20(4): 1298-1311.
- [13] Badri H, Bahreini T, Grosu D, et al. Energy-aware application placement in mobile edge computing: A stochastic optimization approach[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(4): 909-922.
- [14] Poularakis K, Llorca J, Tulino A M, et al. Joint service placement and request routing in multi-cell mobile edge computing networks[C]//IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019: 10-18.
- [15] Farhadi V, Mehmeti F, He T, et al. Service placement and request scheduling for data-intensive applications in edge clouds[J]. IEEE/ACM Transactions on Networking, 2021, 29(2): 779-792.
- [16] Ouyang T, Li R, Chen X, et al. Adaptive user-managed service placement for mobile edge computing: An online learning approach[C]//IEEE INFOCOM 2019-IEEE conference on computer communications. IEEE, 2019: 1468-1476.
- [17] Xu J, Chen L, Zhou P. Joint service caching and task offloading for mobile edge computing in dense networks[C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018: 207-215.
- [18] Tran T X, Chan K, Pompili D. COSTA: Cost-aware service caching and task offloading assignment in mobile-edge computing[C]//2019 16th annual IEEE international conference on sensing, communication, and networking (SECON). IEEE, 2019: 1-9.
- [19] He T, Khamfroush H, Wang S, et al. It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources[C]//2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018: 365-375.
- [20] Antiroiko A V, Valkama P, Bailey S J. Smart cities in the new service economy: building platforms for smart services[J]. AI & society, 2014, 29: 323-334.