

An Adaptive Service Placement Strategy Based on Feature Analysis and Trajectory Prediction in Mobile Edge Computing

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

Pengfan Lu
Faculty of Information Technology
Beijing University of Technology
Beijing, China
lu_peng_fan@emails.bjut.edu.cn

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

Abstract—Edge computing is the complement of cloud computing which provides low-latency and real-time responsive cloud-like services at the edge of the network. Compared with the abundant resources of cloud computing, edge servers have limited resources and heterogeneous, which brings challenges to efficient service placement for multiple mobile users in edge computing. Moreover, the diversity of end devices and the mobility of users also make the placement problem complicate. This paper studies the service placement problem to joint optimize the utilization of edge servers and reduce users' latency. We first introduce a novel multi-level master-slave architecture based on the cloud-edge-device scenario. Based on that, we propose a framework to solve the service placement problem for multi-mobile users. In this framework, we mitigate edge system resource differences and multi-user trajectory unpredictability problems through service feature analysis and user trajectory prediction, respectively. Combining the results of the feature analysis and trajectory prediction, we propose an active service placement method based on deep reinforcement learning. This paper conducts experiments in a simulation environment based on real data sets, and the results show that the algorithm can effectively reduce users' delay under the condition of optimizing edge server utilization.

Keywords—edge computing, service placement, trajectory prediction, reinforcement learning

I. INTRODUCTION

With the rapid development of wireless access technology and the Internet of Things (IoT), massive Internet services and applications are gradually migrating to mobile networks. Due to the widespread access to sensors, large-scale interconnection through IoT devices has been embedded in daily lives. Faced with the surge in demand for mobile access network equipment, the demand for latency-sensitive and compute-intensive data services in mobile networks has surged at an unprecedented rate. Mobile Cloud Computing (MCC) is to use the supercomputing power of cloud servers to perform user data processing tasks in the cloud. MCC is to use the supercomputing power of cloud servers to perform user data processing tasks in the cloud. Although the centralization of cloud computing resources is beneficial to resource management and maintenance, centralized placement leads to long distances between cloud servers and mobile devices. It is difficult to meet the service requests of delay-sensitive applications. As an emerging computing framework, mobile edge computing provides a potential solution to the above challenges. By placing servers at the edge of the network, the

service delay can be greatly reduced, and the cache and computing burden of the data center will be relieved [1].

A. Motivation and Challenges

Since the computing, storage, and bandwidth resources provided by edge servers are often limited, it is difficult to ensure that edge servers are not overloaded after all user service requests by edge servers response. Once the edge server is overloaded, the service latency will increase and the user experience will decrease accordingly. Therefore, some users' service requests still need to be executed on the cloud platform or locally, to ensure the efficient use of edge server resources. How to make decisions on service placement and reduce service delay as much as possible under the constraints of limited resources is one of the most challenging problems in edge computing.

In an edge computing system, any device equipped with computing resources (CPU, GPU, etc.) can be considered a potential edge node, such as routers, small servers, access points, laptops, gateways, etc. User devices can be cell phones, computers, vehicles, sensors, and so on. Therefore, the complex hardware and software environment of edge computing systems increases the difficulty of service placement.

Because of user mobility, low latency and smooth user experience are far from simply pushing cloud servers to the edge of the network. To ensure service continuity when users cross different edge nodes, an effective mobility management scheme should be adopted at the network edge. For example, a vehicle traveling quickly along a road has experienced three edge nodes, following the position of the vehicle by passive placement requires three service migrations. If the service is actively deployed to the intermediate node, the service only needs to be migrated once, which can effectively reduce the service delay.

B. Contributions and Paper Organization

This paper studies the service placement strategy in the multi-user scenario, and aims to reduce user delay under the condition of optimizing edge servers utilization. Our contributions can be summarized as follows:

- We propose a novel multi-level master-slave architecture based on the cloud-edge-device scenario. We divide the edge network into regions that combine one master server and multiple slave servers. The master server is responsible for training the service placement model and giving advises, and the slave servers are responsible for responding to service

Supported by Beijing Natural Science Foundation (4192007), and the National Natural Science Foundation of China (61202076).

requests and executing the placement strategy of services.

- For the proposed multi-level master-slave architecture, we design an edge computing system service placement framework to solve the service placement problem. In this framework, we mainly address the impact of edge system resource differences and multi-user trajectory unpredictability on service placement. Based on the results obtained from feature analysis and trajectory prediction, we propose an active service placement method based on deep reinforcement learning.
- We build a simulation environment based on real data sets and conduct a lot of experiments. The results show that the algorithm can effectively reduce user delay under the condition of optimizing edge servers utilization.

The remainder of this paper is organized as follows. Section II surveys related works. Section III proposes a multi-level master-slave architecture based on a multi-user edge computing scenario, and then formulates the problem model. Section IV designs a service placement framework based on the multi-level master-slave architecture and implements service placement algorithms. Section V includes the experiments. Finally, Section VI concludes the paper.

II. RELATED WORK

In mobile edge computing, when users move to different edge nodes, services need to be migrated to track users to maintain the low latency advantages. User mobility leads to complex trade-offs between service migration costs and long-distance transmission costs. Therefore, it is challenging to obtain the optimal service placement strategy [1-3]. Faced with increasingly complex edge system infrastructure and diverse services [4], focusing on multiple indicators of services can make more effective use of edge server resources. Hao et al. [5] optimize the use of edge system resources by determining in advance the size of each resource allocated by the service through a convex optimization method before service placement.

Ning et al. [6] utilized the Lyapunov optimization method to decompose the long-term optimization problem into a series of immediate optimization problems, using a distributed Markov algorithm to determine the service placement configuration. Gu et al. [7] study a layer-aware micro-service placement and request scheduling at the edge, and an iterative greedy algorithm with a guaranteed approximation rate is designed for service placement. Faced with the impact of user mobility on service placement strategies, these works do not consider the relationship between service performance and operating costs caused by user mobility.

Some works apply reinforcement learning methods to solve the user mobility problem in service placement. Wang et al. [8] formulate the service migration problem with minimum cost as a Markov process. The optimal solution is significantly faster than the traditional method. Ouyang et al. [9] characterized the long-term cost of the system through

Lyapunov optimization and proposed a Markov approximation algorithm to solve the service placement problem under the constraint of a long-term cost budget. Shangguan et al. [10] first proposed an offline algorithm based on dynamic programming to obtain the optimal policy, and then proposed an online microservice coordination algorithm based on reinforcement learning to learn the optimal policy for service placement. However, these works [8-10] do not have predictive information, so latency-sensitive applications are not well handled when the user is moving fast.

Some works reduce the impact of user mobility by introducing prediction methods. Wu et al. [11] proposed a user-centric location prediction method. The algorithm reduces application latency by using the predicted user's next location to realize service migration decisions. Ma et al. [12] adopt multi-steps prediction of trajectory and use the shortest path algorithm to find the best service placement strategy. Although the above methods reduce the delay caused by user mobility through trajectory prediction, they do not consider the impact of edge system resource differences.

Therefore, this paper proposes a service placement framework that takes into account the complex edge computing system environment and the impact on user mobility. Based on service feature analysis and user trajectory prediction, We use deep learning methods to obtain service placement strategies.

III. SYSTEM MODELS

A. Multi-level master-slave architecture

This paper proposes a novel multi-level master-slave architecture based on the cloud-edge-device edge computing system scenario. The overall architecture is shown in Figure 1. The central cloud is located in the top layer, which is pre-installed with all services. It has high-performance on computing, storage, and bandwidth resources that can respond to all service requests from users. In edge layer, we introduce a master server in each region that is used to generate service placement policies. The slave server is responsible for responding to service requests and uploading necessary user information, such as user location. The user layer devices include mobile phones, computers, vehicles, etc. The orange arrows represent the decentralized process of placing services from the cloud to the edge. The dark brown arrows represent a master server sending service placement decisions to slave servers. We use blue arrows to represent service requests that sending from users to edge servers.

User layer devices send out service requests through wireless. Service requests can be performed at edge nodes or central clouds. In this architecture, time is discretized into different time slots. By using virtualization technology, the computing, storage, and bandwidth resources of each edge server are virtualized into independent resources and allocated to users. At the beginning of each time slot, the service request and location information from the user layer is collected by the slave server. The slave server responds to the service requests directly or forwards the information of the ones that cannot be responded to the master edge server.

In Figure 1, the size of resources owned by edge servers is different. The bandwidth, storage, and computing resources possessed by edge servers do not exactly match those required for the service. Black service s_1 pre-installed in the central cloud requires 3 units of communication resources and 2 units of computing and storage resources. Because edge servers have limited resources, it will no longer respond to other service requests after server m_2 placed service s_1 . If three users send requests for services s_1 , s_2 , and s_4 to server m_2 at the same time, s_2 and s_4 are placed to server m_2 . Meanwhile, s_1 will be forwarded to the central cloud, which can optimize edge servers utilization and overall reduce user latency.

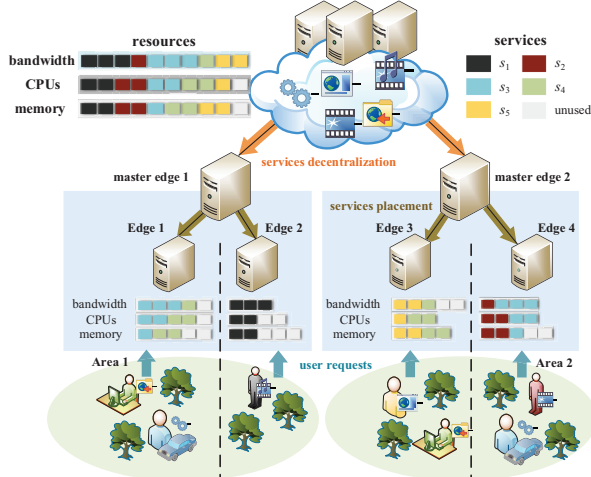


Fig. 1. Multi-level master-slave architecture

B. Network Communication Model

To establish an accurate computing model for the network communication in the edge computing system, the user is first defined, as shown in Definition 1.

Definition 1 (User):User is a quadruple, denoted as $u_i = \langle O_i(x, y), \lambda_i^k, p_i, \delta_i^k \rangle$, where $O_i(x, y)$ is user u_i current latitude and longitude coordinates, λ_i^k is maximum data transfer rate between the user u_i and the edge server m_k , p_i is the signal transmission power of the user u_i , δ_i^k is the channel gain between the user u_i and the edge server m_k . $U(\tau) = \{u_1, u_2, \dots, u_{N(\tau)}\}$ represents the set of users in the τ period in the edge computing environment.

In this system, users and edge servers are wirelessly connected. It is assumed that users use the frequency division multiplexing orthogonal multiple access technology to connect to an edge server to realize bidirectional data transmission, in which the bandwidth of each sub-channel is ϵ . Assuming that the communication interference between each user and an edge server is negligible, the maximum data transmission rate between the user u_i and the corresponding edge server m_k is:

$$\lambda_i^k = n_i^b \cdot \epsilon \log_2(1 + p_i \cdot \delta_i^k \cdot \sigma^{-2}) \quad (1)$$

Here, we use n_i^b to indicate the number of sub-channels allocated to user u_i . σ represents the standard deviation of white gaussian noise in the environment. We list the main notations throughout this paper in Table I for ease of reference.

TABLE I. NOTATIONS

Symbol	Implication
$U(\tau)$	the set of all users in the τ time slot
$M(\tau)$	the set of all servers in the τ time slot
$S(\tau)$	the set of all services in the τ time slot
$O(x, y)$	latitude and longitude coordinates
$\Gamma(x, y)_i^p$	User u_i p-steps trajectory prediction result

C. Service Delay Calculation Model

Since services can be executed on edge servers or the central cloud, two different service delays need to be considered. First, we define the service in the system as follows.

Definition 2 (Service): Services in the edge system are represented as $s_i = \langle r_i, o_i, \epsilon_i, \omega_i, h_i \rangle$, r_i, o_i, ϵ_i represent the total number of computing resources, storage resources and communication resources satisfy to complete the service request, ω_i is the storage required for the service itself, h_i is service resource feature type. $S(\tau) = \{s_1, s_2, \dots, s_{N(\tau)}\}$ represents the set of services in the τ time slot in the edge computing environment, and we assume that a user can only issue one service request at the same time.

Secondly, considering that the difference in hardware resources of edge servers leads to different processing capabilities, edge servers in the system are defined as follows.

Definition 3 (Edge Server): We use $m_k = \{N_k^{cal}, N_k^{stor}, N_k^{comm}, O_k(x, y)\}$ to denote an edge server, $N_k^{cal}, N_k^{stor}, N_k^{comm}$ represent the total number of server unit CPU, memory, and bandwidth resources, respectively. $O_k(x, y)$ represents the latitude and longitude coordinates of the edge server m_k .

Based on the definition of service, following defines the service delay required to meet user service requests in different situations. Then, we give the calculation formula of the total system service delay.

1) Service Latency of Users When Place on Edge Server to Process

When a user's service request is executed on an edge server, a four-part delay needs to be considered. The waiting delay is caused by the user waiting for the wireless channel. The uploading delay is caused by the service input parameters from the user's device to an edge server. The execution delay is caused by an edge server processing service requests. And the backhaul delay is that an edge server return a processed service result. Considering that the data volume of the service result is generally smaller than input parameter data, the backhaul delay is generally negligible. In this system, we also ignore the backhaul delay. In the remaining three parts, the time required for user u_i to upload service s_i input data to edge server m_k can be calculated by the following formula. where $d(u_i, m_k)$ represents the distance between user u_i and edge server m_k .

$$t_i^{up} = \frac{d(u_i, m_k)}{\lambda_i} \quad (2)$$

The calculation formula for the execution delay of processing service s_i on edge server m_k is as follows. n_i^{cal} represents the number of computing resources allocated to user

u_i ; N_k^{cal} represents the computing rate of unit computing resource on edge server m_k .

$$t_i^{cal} = \frac{r_i}{n_i^{cal} \cdot N_k^{cal}} \quad (3)$$

Therefore, if the service request of the user u_i is placed on an edge server for execution, the service delay can be calculated as follows.

$$t_i^{edge} = t_i^{wait} + t_i^{cal} + t_i^{up} \quad (4)$$

Among them, t_i^{wait} is the user u_i waiting delay. The waiting delay is the time interval which a user generate a service request until the service request is processed. In this paper, the service request needs to wait for free wireless channel before uploading. When the request is allocated to the wireless channel, the service request is granted to process. In this paper, t_i^{start} is used to indicate the time when user u_i generate service demand s_i , and t_i^{end} is used to indicate the time when service s_i is allocated wireless channel. Then the waiting delay can be calculated as:

$$t_i^{wait} = t_i^{end} - t_i^{start} \quad (5)$$

2) Service Latency of Users When Place on Cloud Server to process

According to the edge computing architecture, if the user's service request is executed on the cloud server, the service input data first be uploaded to an edge server by the user's device through the wireless channel. Then the edge server forwards this to the cloud server through the wired channel. Considering that the cloud server has strong computing power, the execution delay is negligible compared with the transmission delay. Therefore, when the service s_i is executed on the cloud server, the service delay mainly includes the waiting delay, the uploading delay, and the round trip time (RTT) generated by data transmission between an edge server and the cloud server. The calculation formula is as follows.

$$t_i^{cloud} = t_i^{wait} + t_i^{up} + RTT \quad (6)$$

Since the cloud server is far away from edge servers, the process of an edge server forwarding a service input data to the cloud and the process of the cloud returning the service processed result will generally have a similar delay. The amount of data is irrelevant. Therefore, the RTT can be written as:

$$RTT = \alpha t_{off}^{cloud} \quad (7)$$

t_{off}^{cloud} represents the delay caused by forwarding data from an edge server to the cloud. α is a constant.

3) Total System Service Delay

In this architecture, a service request can be executed at an edge server or the cloud. We use χ_i to represent whether the service is executed at an edge server. If the service is executed on an edge server, then χ_i is 1. Otherwise, the service is executed on the cloud and χ_i is 0. For service s_i , the service delay is:

$$t_i = \chi_i t_i^{edge} + (1 - \chi_i) t_i^{cloud} \quad (8)$$

For all users in the τ time slot, the formula for the total system service delay is as follows.

$$T(\tau) = \sum_{i=1}^{N(\tau)} t_i \quad (9)$$

D. Problem Formulation

In this architecture, we require service placement strategies that minimize the average delay of the system over a long period of time. Therefore, the problem can be modeled as:

$$P1: \min \sum_{\tau=1}^T \sum_{i=1}^{N(\tau)} t_i \quad (10)$$

s.t.

$$\sum_{s_i \in S(\tau)} r_i \leq N_k^{cal}, \forall \tau \quad (11)$$

$$\sum_{s_i \in S(\tau)} \omega_i \leq N_k^{stor}, \forall \tau \quad (12)$$

$$\sum_{s_i \in S(\tau)} \varepsilon_i \leq N_k^{comm}, \forall \tau \quad (13)$$

$$\chi_i \in \{0,1\}, \forall i \in N(\tau), \forall \tau \quad (14)$$

P1 is the objective function and equations 11 to 14 are the constraints. In any time slot, Equation 11 to Equation 13 respectively represent the CPUs, memory, and bandwidth resources allocated by the edge server m_k to users, which cannot exceed the number of resources it has. Equation 14 means that users' service needs must be responded to on the edge server or cloud server. Since this problem is an NP-Hard problem, the algorithmic complexity of finding its optimal solution by traditional methods is exponential. Considering that the process of service placement can be abstracted as a Markov decision process, and the approximate solution can be obtained by using reinforcement learning, so we finally solve this problem based on deep reinforcement learning.

IV. SERVICE PLACEMENT STRATEGY IMPLEMENTATION

There are two key issues for service placement that need to be addressed. The first is the impact of different resources in an edge system. Second, user mobility leads to frequent service migration or long communication distances, which will increase user latency. We propose a service deployment framework to solve the above problems.

A. Service Placement Framework

The framework ensures a complete end-to-end intelligent and automated solution service placement solution. The interactive demonstration process of components in the architecture is shown in Figure 2. In any edge region, there are one master server and multiple slave servers. The master server is responsible for making service placement decisions and communicating with the cloud. The slave server is responsible for responding to specific user requests, collecting and encrypting the user's location information, and sending it to the regional master server. Users' devices can be any IoT

device. The first request from a user is accepted by a slave edge server. If the slave edge server cannot respond to the request, it will notify this information to the regional master server.

In this framework, there are all services placed on the cloud layer by service providers in advance. Before implementing a service placement strategy, the cloud layer will analyze each service resource feature by its historical data, and obtain the typical feature of each service. The cloud layer is also responsible for training the trajectory prediction model of each user. Because users' behavior doesn't change suddenly, the trajectory prediction model is trained by the central cloud and downloaded to use by an edge master server, which can avoid repeated training. The edge master server will regularly exchange user-related information and service features with the cloud layer. The regional master server will train a service placement model and make service placement decisions based on the current regional status information (resource usage of each edge server, user location, etc.), the corresponding user trajectory prediction result, and service features.

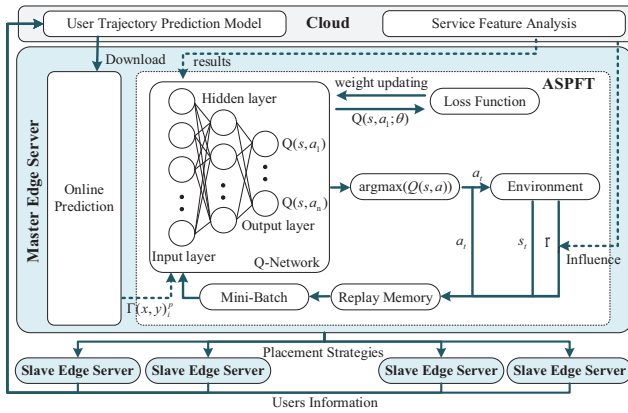


Fig. 2. Service placement framework

B. Analysis of Service Resource Characteristics

In an edge computing system, any device equipped with CPU resources can be considered as a potential edge node, such as routers, small servers, access points, laptops, gateways, etc. User devices can be cell phones, computers, vehicles, sensors, and so on. This leads to resource differences in an edge computing system. This difference makes rational use of edge system infrastructure resources difficult.

Real-life services have many indicators, such as service instance size, request frequency, request duration, request timestamp, request location, and resources (CPU, memory, and bandwidth) required by service instances, etc [3]. In the framework, considering one indicator is difficult to make an appropriate service placement strategy. Therefore, we consider performing feature analysis on real service request data to obtain multiple main indicators. And through the feature analysis results guide the service placement strategy more effectively. The following is the specific process of our feature analysis.

Since it is real service request data [14-16], in the real world, due to various internal or external influences, the absence and abnormal data will occur. To improve credibility and interpretability of the final results, these bad data are culled

and repaired before clustering. For the absence values in the data set, the absence data is filled by the forward and backward moving average method. Let the absence data t_i be the i^{th} data point in the service request data, then the corrected data t_i^* is:

$$t_i^* = \frac{\sum_{h=1}^{|h|} t_{i-h} - \sum_{g=1}^{|g|} t_{i+g}}{|h| + |g|} \quad (15)$$

Among them, t_{i-h} and t_{i+g} represent t_i forward $|h|$ data and backward $|g|$ data, $|h|$ and $|g|$ is generally 5~10.

To more clearly reflect the feature relationship of each indicator in the service request data, the raw data is normalized. The normalization formula is:

$$x^* = \frac{x_i - x_{min}}{x_{max} - x_{min}}, i = 1, 2, \dots, n \quad (16)$$

Among them, x^* and x_i represent the actual value of the i^{th} data and the value after normalization, respectively. x_{max} and x_{min} represent the maximum and minimum values of each metric, respectively. n is the total number of samples.

In this paper, the contour chart method is used to determine the initial k value. The contour value of the i^{th} point on the contour chart is defined as:

$$\beta(i) = \frac{\min(\mathbf{b}) - a}{\max(a, \min(\mathbf{b}))}, i = 1, 2, \dots, n \quad (17)$$

where a is the average distance of the i^{th} point from other points in the same cluster. \mathbf{b} is a vector that represent the average distance of between the i^{th} point and points in different other clusters. The value range of $\beta(i)$ is $[-1, 1]$. The $\beta(i)$ is close to 1, which means that point i^{th} is more inclined to the current class.

In this paper, we analyzed the real dataset [14-16] by clustering. The features of the service are obtained according to the clustering results. For example, if a service has a high request frequency, it means that the service has certain requirements for bandwidth resources. Then the service needs to be matched with an edge server with sufficient bandwidth resources during the service placement process.

Algorithm 1 Offline Service Feature Analysis(OSFA)

Input: Tuples and Sets of edge services \mathbf{S} ;

Output: Mapping of tuples to Sets of edge services \mathbf{S} ;

- 1 Select k and the centroid c_k from \mathbf{S} by contour chart method.
- 2 **For** each service s_i **do**
- 3 choose the nearest centroid
- 4 assign service s_i to the cluster
- 5 **End for**
- 6 **For** each cluster $j = 1 \dots k$ **do**
- 7 new centroid is mean value of all services assigned to that cluster
- 8 **End for**
- 9 Repeat steps 4 and 7 until convergence
- 10 return mapping of tuples to Sets of edge services \mathbf{S} ;
- 11 Select k and the centroid c_k from \mathbf{S} by contour chart method.

Algorithm 1 is pseudocode for service feature analysis. Firstly, we initialize k classes and corresponding center points respectively by the contour chart method. Secondly, we select the nearest center points for each service to join. Then, each class has a mean value as the new centroid. Repeat this process until convergence.

C. DRL-based service placement strategy

According to the clustering analysis of real service requests, we select the CPUs, memory, and bandwidth required by services from many indicators of service requests. In the service placement framework, resources required by the service are relatively fixed, so the analysis result does not need to be updated frequently. Therefore, we let the central cloud complete the content of the feature analysis.

In view of the frequent service migration caused by user mobility or the delay caused by long-distance communication, a user trajectory prediction model is considered to be introduced into the service placement framework to obtain the user's future location information [17]. Since the user trajectory prediction model does not need to be updated and learned frequently, the trajectory prediction model is placed on the central cloud to train. When the edge layer master server needs the trajectory prediction model of a specific user, it will download from the cloud.

Combining the feature analysis and trajectory prediction results, the service placement framework solves the service placement problem through a deep reinforcement learning method. Specifically, we first model the above problem based on a Markov Decision Process (MDP), which includes an intelligent agent, environment, state, action, and reward. The basic idea is as follows: the intelligent agent observes the environment at time slot t and obtains the current state. Then, the agent takes the corresponding action, gives a reward after the environment accepts it, and moves to the next state. An intelligent agent ultimately maximizes the sum of rewards through continuous interaction with the environment.

Therefore, we first need to describe three components based on the service placement scenario: the state space S , the action space A , and the reward function R .

State Space: The master server at the edge layer is responsible for observing each slave servers and user-related information.

$$\Phi = \vartheta(N_k^{cal}, N_k^{stor}, N_k^{comm}), \forall k \in M \quad (18)$$

$$\varphi = \{r_i, o_i, \varepsilon_i, O_k(x, y)\}, \forall i \in N, \forall k \in M \quad (19)$$

$$\xi = \{O_i(x, y), \Gamma(x, y)_i^p\}, \forall i \in N, \forall k \in M \quad (20)$$

$$S = \{\Phi, \varphi, \xi\} \quad (21)$$

Among them, Φ represents the remaining CPU, memory, and bandwidth resources of all slave servers in this area. ri, oi, ε_i in φ are the bandwidth, storage, and computing resources required to serve s_i , respectively. $O_k(x, y)$ is the latitude and longitude coordinates of server m_k where service s_i is placed. ξ include user u_i current latitude and longitude coordinates, and $\Gamma(x, y)_i^p$ represents the p -step coordinates predicted by the user u_i trajectory model.

Algorithm 2 ASPFT

Input: Sets of edge servers \mathbf{M} , users \mathbf{U} , and services \mathbf{S} ;

Output: Service updating decision X of \mathbf{S} in each time slot;

- 1 Offline analysis services feature and get the result A
 - 2 Download the latest user trajectory prediction model from cloud
 - 3 Initialize replay memory D to capacity N
 - 4 Initialize action-value function Q with random weights θ
 - 5 **For** episode=1... M **do**
 - 6 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\varphi_1 = \varphi(s_1)$
 - 7 **For** $t=1 \dots T$ **do**
 - 8 Get prediction trajectory steps Θ and add in φ_t
 - 9 With probability ε select a random action a_t
 - 10 Otherwise select $a_t = \operatorname{argmax}_a(Q(\varphi(s_t), a; \theta))$
 - 11 Execute action a_t in emulator and observe reward r_t and image $x_t + 1$
 - 12 Set $s_t + 1 = s_t, a_t, x_{t+1}$ and preprocess $\varphi_{t+1} = \varphi(s_{t+1})$
 - 13 Store transition $\varphi_t, a_t, r_t, \varphi_{t+1}$ in D
 - 14 Sample random minibatch of transitions $\varphi_t, a_t, r_t, \varphi_{t+1}$ from D
 - 15 Set $y_j = r_j + \gamma \max_a \hat{Q}(\varphi_{j+1}, a; \bar{\theta})$
 - 16 Perform a gradient descent step on $(y_j - Q(\varphi_j, a_j; \theta))^2$ with respect to the network parameters θ
 - 17 Every C steps reset $\hat{Q} = Q$
 - 18 **End for**
 - 19 **End for**
-

Action Space: The action space is a vector. Where $\Lambda(i)$ represents edge server ID where service s_i is placed in the next time slot.

$$A = \Lambda(1), \Lambda(2), \dots, \Lambda(N) \quad (22)$$

Actions taken will generate corresponding rewards, and we will set a baseline algorithm to guide the model to learn in the initial stage. The reward function looks like this:

$$R = \sum_{i=1}^{M(\tau)} t_i - B \quad (23)$$

Where $\sum_{i=1}^{M(\tau)} t_i$ is the estimated total user delay based on the placement decision given by the service placement model, and B is the estimated service delay based on the service static placement.

To minimize the reward function, we first define $Q(s, a)$ as the action value. Then, we can display the expected sum of T timesteps future rewards.

$$Q(s, a) = E(\sum_{t=1}^T \gamma^t R_t | s_t = s, a_t = a) \quad (24)$$

Where $\gamma \in [0, 1]$ is the discount factor and E is the sum of T timesteps future rewards expectation. Therefore, we can formulate the original optimization problem **P1** as finding the

optimal service placement policy, an a^* to minimize the action value $Q(s, a)$.

$$a^* = \operatorname{argmin}_{a \in A} Q(s, a) \quad (25)$$

In this paper, the service placement algorithm is named ASPFT (Adaptive Service Placement Strategy based on Feature Analysis and Trajectory Prediction). The algorithm runs on an edge master server. The pseudocode for this algorithm is shown in Algorithm 2. First, the master server will obtain the corresponding service feature analysis results from the central cloud, and download the latest trajectory prediction model from the central cloud. Then the master server initializes some important parameters related to deep reinforcement learning, including learning rate, memory capacity, learning batch, greedy coefficient and so on. Next, it initializes the environment and trains M times on this (lines 5–19). In each round of iterations, the multi-step trajectory prediction results Θ are obtained through the user’s trajectory prediction model, and the results are added to current state of the environment (lines 8). Then start the training service placement model (lines 9-17). By constantly detecting the environmental status of the edge system, the edge master server can perform continuous training and deliver the placement decisions to slave edge servers.

V. EXPERIMENTS

We build our prototype on a workstation that runs a Linux operating system with I5-12400 CPU, NVIDIA RTX5000 GPU. We used the published Microsoft GPS trajectory dataset [17]. Since this dataset recorded 182 users’ outdoor trajectories in a broad range, we process it according to the features of users’ activities. We first observed the activity tracks of 182 users and marked the longitude and latitude of the origin center coordinates [116.327544, 39.987317]. Then, we take this location as the central point and divide the area with a radius of 2.5 kilometers. We traverse user trajectories to find the ones within this range of area during 60 consecutive time slots. Based on that, we simulate the edge computing network based on U, and we set up 49 edge servers with a range of 450 meters. We range the computing capacity of each server from 2 GHz to 5 GHz, the storage size from 5GB to 10GB, and the communication bandwidth from 300MB/s to 1000MB/s. Compared with the proposed service placement strategy, three baselines are used.

- SP-only (Services Placement Only): The service placement decision is only given by deep reinforcement learning.
- ASP (Services Placement based on Services Feature Analysis): It only uses the service feature analysis results to guide the service placement strategy.
- TSP (Services Placement based on Trajectory Prediction): It only uses the trajectory prediction results to guide the service placement strategy.

A. Service Feature Analysis

We analyze the service request data (computing time, transmission bandwidth, request frequency, transmission data size, etc.) in the real data set [14-16]. In order to facilitate

simulation experiments, we will standardize each resource. Through the cluster analysis of the data, as shown in Figure 3, we final select three categories as service feature indicators.

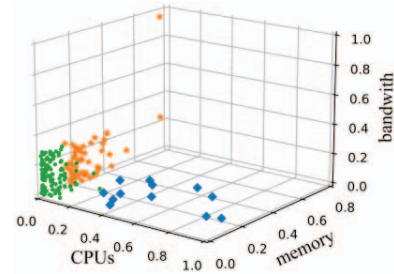


Fig. 3. Service feature analysis

Figure 4 shows the results obtained by guiding service placement through the analysis results. Considering three service indicators during service placement has little effect on the average delay of users, but reduces the number of service migration times. It indicates that the resource usage of edge servers is optimized.

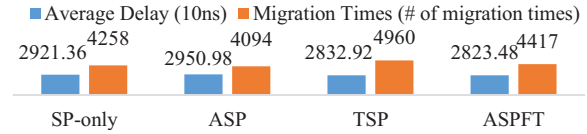


Fig. 4. Service feature analysis under different strategies(20 user)

B. User Trajectory Prediction

In this paper, we choose to maintain a social-LSTM trajectory prediction model for each user on the cloud[18]. We use the real data set Geolife [17] as user trajectory data in the simulation experiment. The average success rate is above 70%, and the results are shown in Figure 5:



Fig. 5. Predict accuracy rate (30 users)

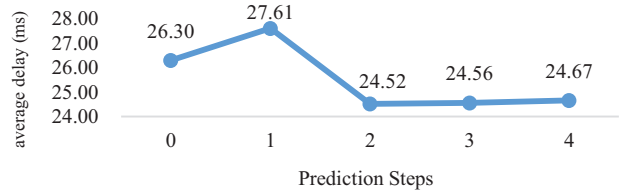


Fig. 6. Multi-step trajectory prediction experiment (15 users)

We conduct simulation experiments on multi-step trajectory prediction, and the results are shown in Figure 6. It can be seen that there will be delay increasing in the one-step prediction. This is because the user will frequently migrate between two edge servers and significantly increase migration latency. In the multi-step prediction, the average delay change is not significant, but the execution time will clearly increase. Therefore, after comprehensive consideration, we finally select a 2-step prediction to guide the service placement strategy.

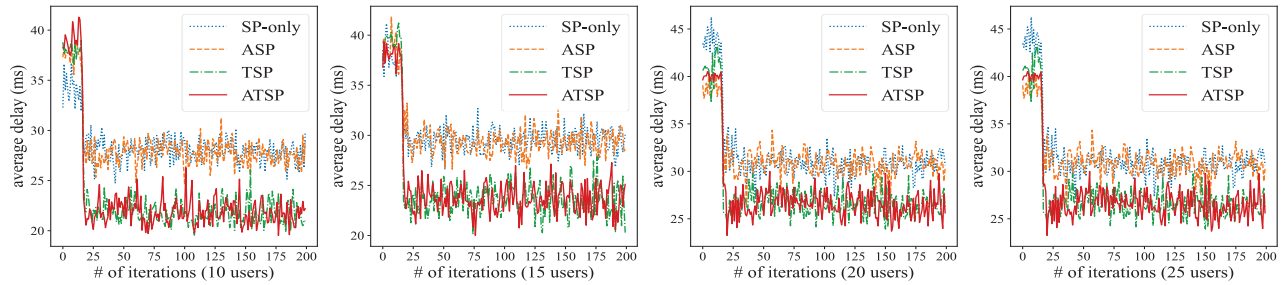


Fig. 8. Convergence average delays under different strategies

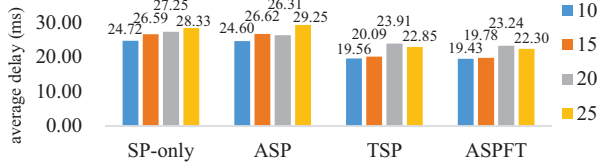


Fig. 7. Average total delays under different strategies

C. Comparison with Comparative Experiments

This paper compares ASPFT with three comparison algorithms, as shown in Figure 7 and Figure 8. With the increase in user number, ASPFT can reduce the average delay of users. Figure 8 shows the results of each algorithm under 200 iterations from 10 to 25 users. It can be found that TSP and ASPFT curves overlap. The reason is that considering multiple service indicators has no significant impact on users' average latency. Multiple service indicators can mainly optimize edge server utilization. The SP-only and ASP curves overlap is the same reason. The results show that the algorithm can effectively reduce user delay under the condition of optimizing edge servers utilization. It can be seen that the ASPFT algorithm outperforms the other three comparison algorithms after a long run time.

VI. CONCLUSION

This paper studies the service placement problem in a multi-user edge computing scenario. We first introduce a novel multi-level master-slave architecture based on the cloud-edge-device scenario. Based on that, we propose a framework to solve the service placement problem for multi-mobile users. Finally, we built a simulation environment based on a real trajectory dataset and conducted a large number of experiments, which proved that the algorithm can effectively reduce user delay under the condition of optimizing edge servers utilization.

ACKNOWLEDGMENT

This work is supported by Beijing Natural Science Foundation (4192007), and supported by the National Natural Science Foundation of China (61202076), along with other government sponsors. The authors would like to thank the reviewers for their efforts and for providing helpful suggestions that have led to several important improvements in our work. We would also like to thank all teachers and students in our laboratory for helpful discussions.

[1] Salaht, F.A., Desprez, F., Lèbre, A. An Overview of Service Placement Problem in Fog and Edge Computing. *ACM Computing Surveys (CSUR)*, 2020, 53, 1 - 35.

[2] Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.*, 2019, 97, 219-235.

[3] Fang, J., Ma, A. IoT Application Modules Placement and Dynamic Task Processing in Edge-Cloud Computing. *IEEE Internet of Things Journal*, 2021, 8, 12771-12781.

[4] Kolosov, O., Yadgar, G., Maheshwari, S., Soljanin, E. Benchmarking in The Dark: On the Absence of Comprehensive Edge Datasets. *HotEdge*, 2020.

[5] Hao, Y., Chen, M., Gharavi, H., Zhang, Y., Hwang, K. Deep Reinforcement Learning for Edge Service Placement in Softwarized Industrial Cyber-Physical System. *IEEE Transactions on Industrial Informatics*, 2021, 17, 5552-5561.

[6] Ning, Z., Dong, P., Wang, X., Wang, S., Hu, X., Guo, S., Qiu, T., Hu, B., Kwok, R.Y. Distributed and Dynamic Service Placement in Pervasive Edge Computing Networks. *IEEE Transactions on Parallel and Distributed Systems*, 2021, 32, 1277-1292.

[7] Gu, L., Zeng, D., Hu, J., Li, B., Jin, H. Layer Aware Microservice Placement and Request Scheduling at the Edge. *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 1-9.

[8] Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K.S., Leung, K.K. Dynamic Service Migration in Mobile Edge Computing Based on Markov Decision Process. *IEEE/ACM Transactions on Networking*, 2019, 27, 1272-1288.

[9] Ouyang, T., Zhou, Z., Chen, X. Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing. *IEEE Journal on Selected Areas in Communications*, 2018, 36, 2333-2345.

[10] Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., Shen, X. Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach. *IEEE Transactions on Mobile Computing*, 2021, 20, 939-951.

[11] Wu, Q., Chen, X., Zhou, Z., Chen, L. Mobile Social Data Learning for User-Centric Location Prediction With Application in Mobile Edge Service Migration. *IEEE Internet of Things Journal*, 2019, 6, 7737-7747.

[12] Ma, H., Zhou, Z., Chen, X. Leveraging the Power of Prediction: Predictive Service Placement for Latency-Sensitive Mobile Edge Computing. *IEEE Transactions on Wireless Communications*, 2020, 19, 6454-6468.

[13] Council, S. P. Spc trace file format specification. 2002.

[14] Ambati, P., Irwin, D.E., Shenoy, P.J. No Reservations: A First Look at Amazon's Reserved Instance Marketplace. *ArXiv*, 2020, abs/2005.12249.

[15] Ambati, P., Bashir, N., Irwin, D.W., Shenoy, P.J. Waiting Game: Optimally Provisioning Fixed Resources for Cloud-Enabled Schedulers. *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, 1-14.

[16] Rudenko, A., Palmieri, L., Herman, M., Kitani, K.M., Gavrila, D.M., Arras, K.O. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 2020, 39, 895 - 935.

[17] Zheng, Y., Zhang, L., Xie, X., Ma, W. Mining interesting locations and travel sequences from GPS trajectories. 2009, WWW '09.

[18] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S. Social LSTM: Human Trajectory Prediction in Crowded Spaces. 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 961-97