Mobility-aware and Privacy-protecting QoS Optimization in Mobile Edge Networks

Huiying Jin, Pengcheng Zhang, *Member, IEEE*, Hai Dong, *Senior Member, IEEE*, Xinmiao Wei, Yuelong Zhu, and Tao Gu, *Senior Member, IEEE*

Abstract—With the rapid development of 5G technologies, the demand of quality of service (QoS) from edge users, including high bandwidth and low latency, has increased dramatically. QoS within a mobile edge network is highly dependent on the allocation of edge users. However, the complexity of user movement greatly challenges edge user allocation, leading to privacy leakage. In addition, updating massive data constantly in a dynamic mobile edge network also crucial to ensure efficiency. To address these challenges, this paper proposes a dynamic QoS optimization strategy (*MENIFLD_QoS*) in mobile edge networks based on incremental learning and federated learning. *MENIFLD_QoS* optimizes service cache in edge regions and allocates edge servers to edge users according to the locations of edge servers accessed by edge users in mobile scenarios. While optimizing regional service quality, the system can effectively protect user privacy. In addition, for dynamic incremental data, *MENIFLD_QoS* trains updated data based on the strategy of incremental learning hence significantly improves optimization speed. Experimental results on an edge QoS dataset show that the proposed strategy achieves global optimization in both multi-variable and multi-peak user allocation scenarios and notably enhances the training efficiency of the regional invocation model.

Index Terms—Mobile edge, Quality of Service, Incremental Learning, Federated Learning, Mobility Aware, Edge User Allocation.

1 Introduction

Web services are loosely-coupled, self-contained and platform-independent applications to implement Service-Oriented Architecture [1]. Web services abstract diverse functional components of applications or programs into reusable services and enables interoperability and composability through provisioning standard interfaces. With the dawning of the Internet of Everything (IoE) era, Internet users produce a huge volume of data in the network, where hundreds of different web services (e.g., webpages visiting, video watching, file uploading and downloading, etc.) are deployed at the edge to consume these data [2]. Given that many web services possess similar or same functions, service users need to select appropriate services to meet their demands [3]. For example, when edge service providers provide video services in exactly same themes, users may be inclined to choose the one with higher resolution. Web service selection highly relies on the non-functional properties of web services, including response time, and throughput which are termed quality of services (QoS).

Nowadays 5G technologies have been involved in many countries' future strategic development plans [4]. 5G stations are the central of 5G network, they enable signal transmission between wired communication networks and wireless terminals, which are predicted to be deployed in high densities [5]. As one of the core technologies of 5G, edge computing sinks high-bandwidth, low-latency, and localized services to the edge of network [2], [6]. It essentially addresses the problems of network congestion and service response delay. It also supports real-time and bandwidth-intensive services in 5G networks. However, data privacy and security gradually become users' top concern while users enjoy the convenience brought by 5G technologies [7]. For example, in 22 September 2022, Optus, the second largest telecommunications company in Australia, suffered from a significant cyberattack, resulting in a leakage of over 2.1 million customers' private information, such as passports, driver licenses, etc [8]. This has been the eleventh data breach occurred in Australia in 2022, each of which affects at least 10,000 customers [9]. Therefore, it is critical to protect privacy and security while providing intelligent services and guaranteeing QoS.

In the process of service sinking, a user needs to be allocated to one of his/her accessible edge servers to be able to access these sunk services considering the resource capacity of these servers [10]. The distance and the capacity of the allocated server may influence quality of services. For example, when computing tasks of web services are allocated to edge servers, the speed of data transfer fluctuates with the varied distance between users and servers, and the computing time is shifted with different capacity of servers [11]. Therefore, the differences in computing capacity of edge servers and the dynamic locations of user devices lead to different QoS according to different user allocation strategies. Many QoS optimization studies have been carried out in the domain of edge computing. Researchers consider various constraints such as edge servers' resource consumption, service business logic and user mobility to

- *H. Jin, P. Zhang, X. Wei and Y. Zhu are with the Key Laboratory of Water Big Data Technology of Ministry of Water Resources* and *the College of Computer and Information, Hohai University, Nanjing, China*
  E-mail: 367046895@qq.com; pchzhang@hhu.edu.cn; wxm2333@outlook.com; ylzhu@hhu.edu.cn
- *H. Dong is with the School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia*
  E-mail: hai.dong@rmit.edu.au
- *T. Gu is with the Department of Computing, Macquarie University, Sydney, New South Wales, Australia*
  E-mail: tao.gu@mq.edu.au

Manuscript received XXX XX, XXXX; revised XXX XX, XXXX.

formulate task scheduling strategies. In addition, multiple QoS attributes such as response time and throughput are taken into account to find optimal scheduling strategies. Contemporary studies mainly focus on task acceptance rate optimization [12], strategy cost optimization [13], [14], task offloading optimization [11], [15] and resource allocation optimization [16], [17]. The existing studies focus on relatively static scenarios [11], [16], [18]. In contrast, realistic user movement scenarios are inconstant, where more dynamic factors such as user trajectories and network conditions need to be considered. The existing studies are thus unsuitable for these realistic scenarios.

As shown in Fig. 1, a mobile edge network (MEN) comprises multiple layers, including the layers of mobile devices, mobile edge computing and Internet cloud from bottom to top [19]. Cloud servers sit on the Internet cloud layer. Edge servers such as base stations and edge servers sit on the mobile edge computing layer. The mobile devices layer has a collection of mobile devices such as smartphones, laptops, and wearable devices. MEN integrates the computing capabilities of the edge network into the mobile network architecture. In a MEN, base stations are the communication nodes of network, and edge servers are the computing nodes of network. Since edge servers are commonly deployed in based stations [20], we assume that a base station and an edge server are in the same location. In a MEN, mobile devices face the risk of private data leakage in the process of edge service invoking [21]. The privacy data of mobile devices mainly include user locations and user features (i.e., users' favorite services, QoS values, and historically invoked service records, etc.). When optimizing edge service quality, user locations are needed to predict trajectories of mobile devices [13]. This breaches user location privacy. When optimizing edge caching, a user's feature information needs to be adopted to train the user's private models [17]. This causes user feature data leakage. Therefore, privacy protection is vital to edge optimization.

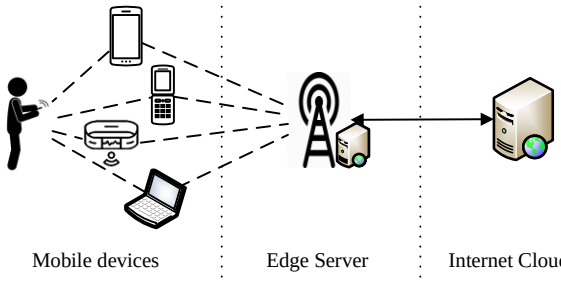


Fig. 1: Architecture of MEN

We illustrate the aforementioned problems using the following scenario. Assume that user *Vae* and *Lee* are in a real-time communication via a MEN. As illustrated in Fig. 2, *Vae* and *Lee* are both moving from region 1 to region 3. In region 1, *Vae* and *Lee* maintain communication by respectively accessing edge servers S_1 and S_2. They both head west through region 2 while maintaining the communication service provisioned by edge servers S_3 and S_4, respectively. Now the question is how to allocate edge servers to *Vae* and *Lee* to ensure communication quality and acquire higher quality of communication in region 3. It needs to forecast their trajectories, allocate edge servers, and deploy communication service in advance.

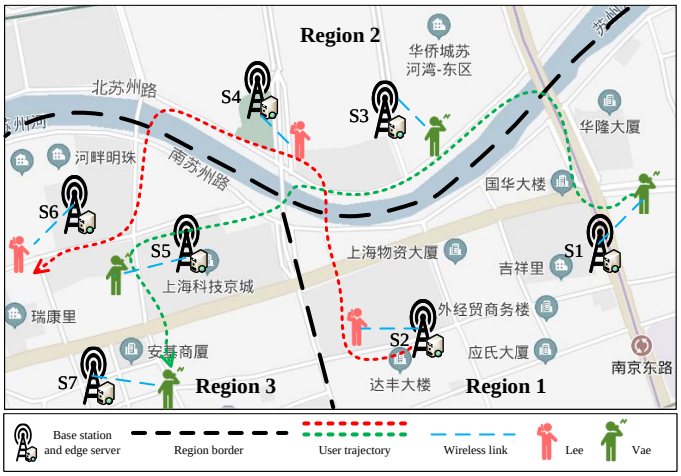


Fig. 2: Scenario of service invocation in a MEN

Existing optimization works deploy and schedule communication services by directly adopting the moving paths of users to enhance the quality of communication services without considering privacy preservation in optimization. When forecasting the trajectory of *Vae*, his current location will be analysed to predict his future location. It leads to location privacy leakage. To optimize edge server cache, each server needs to analyse users' demanded services and QoS according to user features. Therefore, these user features are required for training regional service invocation models in edge servers. However, the adoption of user features causes privacy leakage.

Several challenges exist in MEN based Web service QoS optimization.

i). Ignorance of user information leakage during QoS optimization. As a large number of users invoking edge services, more user information will be used to train user invocation models (i.e., indicating users' service preferences) and user mobility models (i.e., reflecting users' historical trajectories) for user allocation. Hence, it is necessary to adopt users' historical behavior features to analyse users' service invocation and predict location information in a MEN. This process leads to the leakage of user feature and location information. Therefore, privacy protection is of great significance in QoS optimization.

ii). Explosive training cost caused by dynamic training of incremental data. Continuously invoking edge services by mobile users quickly increase the amount of invoking data. This dramatically boosts the training cost of edge region caching models. The increased training time exceeds the time constraint of an optimization model for processing a batch of data, which triggers deadlock once new data arrives. It paralyses optimization algorithm in late stages of optimization. Therefore, reducing the cost of incremental data based model training is crucial to the effectiveness of optimization.

iii). Designing a comprehensive and multi-faceted QoS optimization solution is significant but neglected. During the process of QoS optimization, the main optimization aspects include: user allocation, service task scheduling and edge

caching, etc. All the factors have been proven to be efficacious and critical in edge QoS optimization. Nevertheless, no effort has been attempted to effectively integrate these factors. Therefore, it is of significant value to propose a comprehensive and multi-faceted QoS optimization strategy in MENs.

To address the aforementioned problems, this paper proposes a dynamic QoS optimization strategy based on incremental learning (IL) and federated learning (FL) in MENs (MENIFLD_QoS). We aim to accurately predict users' movement and analyse service invocation behavior in the complex and dynamic MEN. The proposed strategy trains users' invocation preference models and regional public models to determine edge service cache through federated learning [22]. It is able to shorten the response time of edge services while protecting user feature information. The strategy preserves users' location privacy based on the concept of "K-Anonymity". It can also restore users' mobility information via training a user mobility model with locations of edge servers accessed by users. The restored mobility information is then adopted to optimize the allocation of edge servers for edge users. In addition, we adopt IL to reduce the growing training cost caused by constantly incremented data. IL can preserve most of the previously learned knowledge, in addition to learning new knowledge from incoming data. In this way, training speed can be greatly improved. Finally, we design an improved artificial bee colony algorithm to properly allocate users to edge servers and optimize service quality in edge region. The major contributions of this paper are summarized as follows.

- We leverage the idea of FL and "K-Anonymity" concept to protect user feature privacy and location privacy during the optimization process. We adopt a unified user model delivery paradigm based on FL, upon which all users employ the invocation models with same dimensions to interact with edge servers. It makes each individual model difficult to be distinguished and effectively protects user feature privacy. We generalize user-specific information based on the concept of "K-Anonymity". Lagrangian interpolation is then applied to locations of edge servers accessed by users to restore users' mobility information, while effectively protecting user location privacy.
- We adopt IL for incremental data based service invocation model training. IL does not need to repeatedly process historical data and can keep the old effective knowledge when learning new knowledge from incremental data. The experimental results show that the training time of our service invocation model is saved by 75.8% using IL.
- We design an optimization strategy based on IL and FL to optimize the overall QoS in an edge region from multiple facets: quality for edge services, service caching for edge servers and server allocation for edge users. The experimental results prove that our strategy can improve the overall QoS compared to existing strategies while realizing privacy protection and enhancing training efficiency.

The structure of the paper is organized as follows. Section 2 surveys state-of-the-art QoS optimization approaches and mobile edge technologies, and discusses their limitations. Section 3 introduces the dynamic QoS optimization strategy based on IL and FL in a MEN. Section 4 analyzes the experimental results based on several data sets. Section 5 concludes the paper and plans our future work.

2 RELATED WORK

The existing service layer optimization work in a mobile edge environment mainly includes user allocation optimization, task scheduling optimization, and MEN computing resource and QoS optimization.

2.1 Mobile user allocation

In the mobile edge environment, edge servers can be accessed by users when the users are under the servers' coverage and the servers' resources are not full utilized. Since the storage and computing capacity of users' mobile devices is limited, intensive computing tasks need to be shared among adjacent edge servers to achieve workload balance [23]. The overall QoS (i.e., quality of all the services provided) in an edge region is closely related to how edge users are allocated to that region [10]. Therefore, the overall QoS can be viewed as an effective indicator to measure the quality of edge user allocation and edge resource scheduling.

He et al. [23] define this type of problem as an Edge User Allocation (EUA) problem. They analyze the constraints of resources and distances in the edge environment to model such a problem and propose heuristic methods to solve the EUA problem. They [10] further consider the dynamic QoS levels of edge service users to find a solution that can maximize the overall QoE of application users. EUA is performed based on distance perception and confrontation perception. Peng et al. [24] make online EUA decisions based on users' mobility and time-varying positions. The above-mentioned EUA problem does not consider the overall QoS optimization for an edge region.

2.2 Task offloading and scheduling

A large amount of existing edge computing optimization work focuses on user offloading decision-making problems. This type of work first estimates the delay and energy consumption of a computing task. It then develops a scheduling strategy for deciding deploying the task to the server or executing it locally. Whether the energy consumption and delay are optimized after the implementation of the strategy is then observed.

Deng et al. [16] propose to reduce deployment costs and improve quality of services like response time by optimizing application deployment in the mobile edge environment, when computation resource and storage resource are limited. Miao et al. [15] propose a computation offloading and task migration algorithm based on task prediction, which effectively reduces the total task delay with increasing data and services. The mobility of users in the edge environment is also a factor that cannot be ignored. In the medical monitoring scenario, Xu et al.'s work [11] can reduce energy consumption and workload by adjusting mobile-aware task offloading and scheduling strategies to best meet the flexible

requirements of distributed environments and real-time response of various services. Wang et al. [13] also consider task attributes, user mobility and network constraints to reduce the delay in task executions caused by task scheduling.

2.3 Edge caching

Considering that service placement on edge servers nearby can reduce the latency of users to access services, many researchers have explored optimal service placement schemes to maximize the QoS on mobile devices. In recent years, how to cache services on edge servers has attracted the attention of many scholars. Liang et al. [25] propose to place service entities with user states and computation tasks in distributed interactive applications to achieve low-delay pairwise interactions under the budget constraint. You et al. [26] propose a service placement model that measures cost and delay. Qian et al. [17] propose a service placement strategy based on FL combined with user preferences, while protecting the sensitive historical data of users' mobile devices. Wang et al. [27] train DRL agents in MEC systems to optimize edge caching and computing. They introduce FL for optimized intelligent resource management. However, this approach only considers the cache content.

The existing work share the following common issues. First, the existing work only focuses on addressing the QoS optimization problem from a single perspective, i.e., user allocation, service task scheduling or edge caching. There is a lack of exploration from all of the aforementioned aspects. Next, none of the above QoS optimization work considers how the environmental changes resulted from users' mobility impact task offloading and scheduling as well as service caching. Finally, no solution is proposed for preventing the leakage of user feature privacy and location privacy. There is currently no dynamic QoS optimization method considering privacy protection and user mobility in MENs.

3 THE MENIFLD_QOS STRATEGY

The threat model and privacy protection optimization scheme is presented in Section 3.1. The overview of the MENIFLD_QoS strategy is outlined in Section 3.2. The three steps of MENIFLD_QoS are introduced in details in Section 3.3, Section 3.4 and Section 3.5.

3.1 Threat Model and Privacy-protecting Scheme

QoS optimization in the edge environment usually targets three components: user groups on mobile edge networks, edge server clusters, network infrastructure providers and edge service providers. As shown in Fig. 3, *Vae* and *Lee* are communicating via a mobile edge network. When they initiate the communication through an edge service (e.g., WhatsApp), the network infrastructure provider and the edge service provider can identify their geographic location, communication time, service types and other information in real time. The providers authorize the edge server cluster to utilize the user and service information. The cluster trains a user feature model based on the information to realize user personalized service caching and optimization. For attackers, their goal is to steal sensitive user information and obtain user features. During this process, attackers can steal user privacy through the following channels: 1) during the user and service information authorization, attackers can steal user sensitive information (e.g., *Vae* and *Lee*'s geolocation, communication time, etc.) by forging the addresses of trusted edge server clusters; 2) during the user feature model training, attackers can identify or infer user model parameters (e.g., service types *Vae* and *Lee* invoked) by inducing them to authenticate (e.g., constructing a fake "login" module). Therefore, a scheme is urgently needed to protect user privacy.

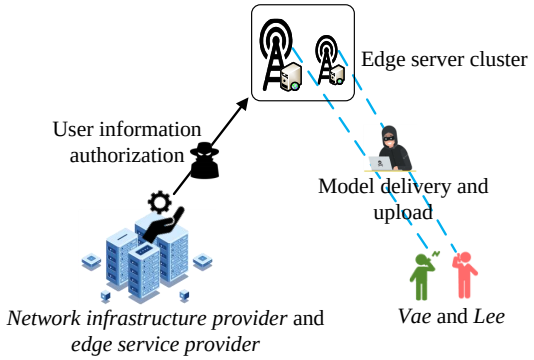


Fig. 3: A privacy threat scenario during edge QoS optimization

When we visit some websites/applications (e.g., real estate websites, etc.), we inevitably receive some pop-up advertisements provided by third parties. The locations of the advertised product sellers or service providers are usually close to ours. This is a typical example of information leakage via channel 1). In addition, the product/service types on the pop-up advertisements are usually relevant to the ones we are browsing in the current website. This is an example of information leakage through channel 2).

The existing privacy protection QoS optimization studies in mobile edge computing are mainly based on protection of user locations and other privacy-sensitive information. They usually rely on fuzzy functions [12] or encryption protocols [28] to protect users' locations. However, most of these techniques require the input of users' locations, which are vulnerable to attacks or theft during user information authorization. To address this issue, a federated learning-based model [17] is proposed, which aims to protect sub-model parameter transfer. However, these parameters can still be leaked by analyzing the differences between user sub-models during the parameter transfer. Therefore, how to protect user privacy is a significant issue in edge QoS optimization.

We realize privacy-preserving edge QoS optimization through user location protection and model unification. Fig. 4 shows our proposed edge privacy protection optimization scheme, where MENIFLD_QoS is employed for QoS optimization. Here users are data producers. They generate relevant information (e.g., user locations and service attribute values, etc.) during their movement and service invoking processes. The mobility model is trained by the edge server clusters. When network infrastructure providers and edge service providers authorize edge server clusters to access the

user information, the concept of "K-Anonymity" is adopted to protect user location privacy. "K-Anonymity" processes the original data before the release of data by means of generalization (i.e., data description abstraction) to protect individual privacy [29]. Its principle is to weaken the association between private information and personal identities. Fig. 5 shows an example of the user information generalization process based on the concept of "K-Anonymity", in which *access time* and *invoked service* are generalized. Similarly, a user's exact location (i.e., *user location*) at a moment is generalized by replacing it with the location of an edge server accessed by this user at that moment. In this regard, the exact locations of any user in the final processing result are hidden, and the user location privacy is effectively protected. Once network infrastructure providers and edge service providers authorize edge server clusters to access the generalized information, the clusters first identify the sequence of edge servers accessed by the same user based on the *IMEI* (e.g., 356709081343630), and then utilize the edge location information (e.g., longitude: 118.793851, latitude: 31.9181716) to train the mobility model to predict the user's movement direction. When mobile users interact with edge servers, they download a unified model as the foundation to train personalized models, and submit a set of parameters covering the entire model (i.e., part of the parameters are users' individual training parameters, and the remaining are 0s, thereby all users' model dimensions are same, making each individual model difficult to be distinguished) during the upload process to hide individual features. The scheme effectively alleviates the problems faced by the existing approaches in the privacy threat scenario.

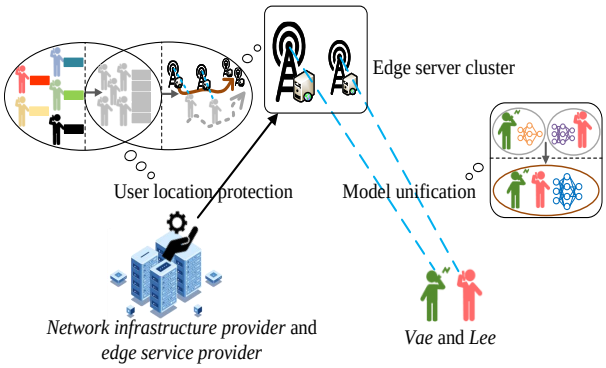


Fig. 4: Privacy protection based edge QoS optimization scheme

3.2 Overview of MENIFLD_QoS

Optimizing the overall QoS (i.e., quality of all the services) in an edge region is closely related to the proper allocation of users to the edge servers covering that region [10]. Therefore, we aim to obtain the optimal overall QoS of each edge region through edge user allocation optimization.

We propose a dynamic service quality optimization strategy MENIFLD_QoS based on IL and FL in a mobile edge environment. IL can avoid boosted training costs caused by repeated training upon incremented historical data. To prevent user privacy data leakage during the training process,

IMEI	The location of accessed edge server	Access time	Invoked service	User location
356709081343630	(118.793851, 31.918716)	15:10:04	YouTube	(118.793783, 31.918946)
354110231685545	(118.793851, 31.918716)	15:10:36	Metacafe	(118.793851, 31.918705)
866014046265140	(118.793851, 31.918716)	15:11:28	Vimeo	(118.793954, 31.918896)
355716071621199	(118.793851, 31.918716)	15:11:35	Twitch	(118.79359, 31.91872)
356709081343630	(118.78878, 31.921046)	15:12:17	YouTube	(118.78869, 31.921613)
866014046265140	(118.78878, 31.921046)	15:13:18	Vimeo	(118.789373, 31.920172)
864358043080253	(118.78878, 31.921046)	15:13:46	MySpace	(118.789355, 31.921092)
...	...	...	...	...

IMEI	The location of accessed edge server	Access time	Invoked service	User location
356709081343630	(118.793851, 31.918716)	15:10-15:12	video	(118.793851, 31.918716)
354110231685545	(118.793851, 31.918716)	15:10-15:12	video	(118.793851, 31.918716)
866014046265140	(118.793851, 31.918716)	15:10-15:12	video	(118.793851, 31.918716)
355716071621199	(118.793851, 31.918716)	15:10-15:12	video	(118.793851, 31.918716)
356709081343630	(118.78878, 31.921046)	15:12-15:14	video	(118.78878, 31.921046)
866014046265140	(118.78878, 31.921046)	15:12-15:14	video	(118.78878, 31.921046)
864358043080253	(118.78878, 31.921046)	15:12-15:14	video	(118.78878, 31.921046)
...	...	...	...	...

Fig. 5: User information processing based on the concept of "K-Anonymity"

we adopt 1) FL to train service invocation models and 2) "K-Anonymity" concept and Lagrangian interpolation to train the user mobility model. The process is shown in Fig. 6.

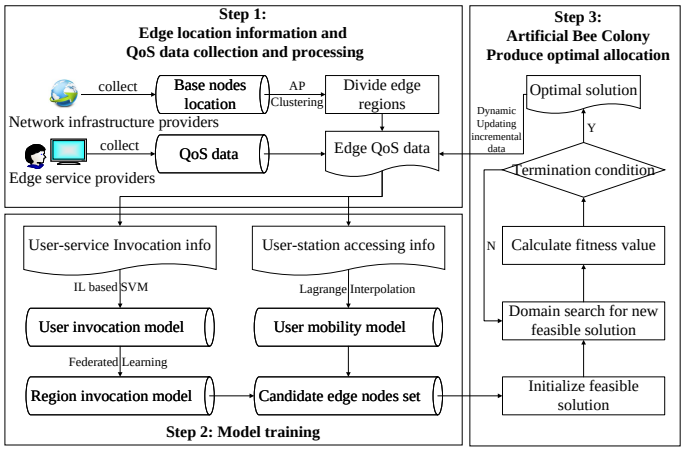


Fig. 6: The overview of MENIFLD_QoS

An edge user in a MEN continuously generates new location and service invocation data. The new data provides foundation for training an invocation model that indicates the user's preference and a mobility model that predicts the user's direction. Following the principle of IL, MENIFLD_QoS has a memory function for learned knowledge, which does not need to retrain historical data to effectively learn new knowledge from new data. An IL based adaptive framework built on SVM is adopted to dynamically adjust the user invocation models to reduce the training cost for the boosting data, so as to reduce the cost of model training and improve optimization efficiency.

The network infrastructure providers and edge service providers respectively own the locations of edge servers accessed by users and the quality of the services invoked by users from the accessed edge servers in each time period [30]. When users move, communicate, and invoke services,

network infrastructure providers and edge service providers can obtain the information about these activities. The location of edge nodes are divided according to edge regions. Users' movement data is acquired from the data sets by extracting the locations of edge servers accessed by the users. Each user's movement directions are then predicted based on their mobility models trained upon their historical movement data. A user's service invocation data in all edge servers of a region is used to train the user invocation model. A public model is trained for each region to reflect all the users' service invocation preference in that region. This public model will be employed to determine the service caching in the edge servers of that region. It aims to optimize the response time of edge services by providing the services with lower transmission data volume and higher response speed. Finally we employ a heuristic method to allocate the edge users to optimize the overall QoS in an edge region. The whole process comprises the following three primary steps:

i). Edge location information and QoS data set collection and processing. The first step is to respectively collect and process edge location information and QoS attribute values. The collected edge location information contains users' edge server access records. The QoS data set includes response time and throughput of services invoked by users from the edge severs in each time slice. First, we map the latitude and longitude values of the edge servers to a Cartesian coordinate system to reflect the actual distance between the edge servers, so as to obtain the distribution of edge servers and users' edge server access records. The latter includes the locations of the edge servers accessed by the users and their access time, etc. Next, we allocate the QoS data of the services invoked by the users from edge servers to the corresponding users and edge servers to form an edge QoS data set.

ii). Model training. After obtaining the edge QoS data set, we extract the user feature data (i.e., services invoked by users) in terms of user id. The user feature data is to train a gradient descent logistic regression model (i.e., the user invocation model), for each user to indicate their service preference. Finally we use FL to extract the parameters of the user invocation models in an edge region to train a regional public service caching model. FL can effectively protect user privacy during the model training. The regional public service caching model is further trained based on the principles of IL and FL for the newly generated service invocation data. In addition, we employ Lagrangian interpolation to predict the user's movement direction according to the locations of edge servers accessed by the user. The predicted user movement direction is used to determine the candidate edge server set to be accessed by the user. Moreover, the newly accessing edge location information of a user can dynamically update the predicted movement direction to regenerate the candidate edge servers set.

iii). Regional dynamic QoS optimization. On the basis of the service invocation information and candidate edge server set of a user at the next moment, an edge server is allocated to the user. Each allocation is regarded as a feasible solution for the EUA problem. We then calculate the fitness value, that is, the overall QoS value (i.e., the quality of all the invoked services in an edge region) of each feasible solution. Finally, an artificial bee colony algorithm is used to iteratively generate and update the feasible solutions to find the optimal solution.

3.3 Dynamic Data Collection and Preprocessing

The aforementioned scenario shows that mobile users *Vae* and *Lee* simultaneously access a real-time communication service during their movement. When a user connects to an edge server via wireless networks, the network infrastructure providers will obtain the information about the user's location. However, the network infrastructure providers are only the data owners and providers, and they do not perform model training and other tasks. *Vae* accesses the communication service from edge servers S_1, S_3, and S_5 respectively, while *Lee* accesses the service from edge servers S_2 and S_4. This process generates some access records, including ids and locations of the accessed edge servers, user ids and server access time (e.g. $\{S_1$, *loc*(S_1), u_{Vae}, $T_1\}$), and QoS data (e.g. response time, throughput) of the accessed service from different servers.

The data collection and preprocessing procedure collects and processes the aforementioned access records and QoS data respectively [31], and clusters the geographically close edge servers that share the similar edge network environment [32].

The edge region division process is to narrow down solution space and improve efficiency for the forthcoming optimization. A geographical area is divided into several edge regions according to locations of the edge servers in a data set. This process comprises the following three steps.

Step 1: We screen out active edge servers according to all users' access records in the area, and extract all non-redundant edge servers' locations according to their IP addresses.

Step 2: We employ Universal Transverse Mercator Projection (UTMP) to project the latitude and longitude of each edge server into the corresponding Cartesian coordinates to accurately measure the actual distance between the edge servers. UTM is broadly used as a reference projection system to compile maps [33].

Step 3: The Affinity Propagation (AP) clustering algorithm is used to cluster the edge servers [34]. Compared with other clustering methods, the AP clustering algorithm can handle large-scale data with better clustering performance and higher efficiency. The algorithm is introduced as follows:

Given $N = \{n_i\}_{i=1}^{K}$ is the set of edge server locations to be clustered, where K represents the number of edge servers, and S is a matrix that describes the similarity between the locations, for any location n_k in the set N, the attraction information $r(i,k)$ and attribution information $a(i,k)$ can be used to characterize the location. The attraction information $r(i,k)$ describes the degree of attraction of a point k as the cluster center to a point i. The attraction information $r_{t+1}(i,k)$ is iterated as follows:

$$r_{t+1}(i,k) = s(i,k) - \max_{k \neq k'} \{\alpha_t(i,k') + s(i,k')\} \quad (1)$$

where $s(i,k)$ represents the similarity between the point i and the point k in the matrix S. The attribution information $a(i,k)$ describes the adaptation degree for the point i

selecting k as the cluster center. The attribution information $a_{t+1}(i,k)$ is iterated as follows:

$$\alpha_{t+1}(i,k) = min\{0, r_t(k,k) + \sum_{i' \notin \{i,k\}} max\{0, r_t(i',k)\}\}, i \neq k \quad (2)$$

$$\alpha_{t+1}(k,k) = \sum_{i' \notin \{i,k\}} max\{0, r_t(i',k)\} \quad (3)$$

After initialization, the value of $a(i,k)$ will gradually decrease to reach a stable state along with the increase of the $r(i,k)$ value. The above steps are iterated until the decision of the clusters remains unchanged or varies in a small range, or the number of iterations reaches a predefined number.

3.4 Model Training

We use IL to train the increasing user invocation data as the user invocation model. Then use FL to train the public model for each edge region, determining the service cache. At the same time, we use the user-server accessing data to train the user mobility model to perceive the mobility of the MEN.

3.4.1 User Invocation Model Training

We only use user feature information (i.e., services invoked by users) to train the user invocation model. This process does not involve users' sensitive information (i.e., users' specific locations).

SVM is a powerful machine learning tool broadly applied for classification and regression. SVM also has remarkable generalization performance in sparse high-dimensional environments. The data learning theory proposed by Vapnik provides theoretical support for SVM to reduce structural risks [35]. We use SVM to train the user invocation model to optimize service caching for each edge server. To train the SVM, we collected and computed 3 static features from user samples. With the help of these features and Gaussian kernel with variance $1/(2 * scale)$, we can obtain the main parameters for SVM. Therefore, the invoked services are checked by the trained SVM classifier, and We can divide the training examples into three different categories based on the partial derivatives g_i: a margin support vector set S ($g_i = 0$), an error vector set ε ($g_i < 0$) and a reserve vector set R ($g_i > 0$). Traditional SVM can help select the most frequently invoked services at a time period, but is not enough to train the boosting samples in the long term. We deploy IL based SVM to train new samples dynamically.

When the learning sample set receives new invoking data, we use IL to train the samples with the goal of preserving the KKT conditions for all the previous training data [36]. We need to vary the margin vector coefficients in response to the perturbation imparted by the incremented new coefficients to maintain the KKT conditions.

Our objective is to determine the necessary changes in the margin vector coefficients $\Delta\alpha_k$ and the bias Δb that preserve the KKT conditions on all learned data according to a given perturbation on partial derivatives of dual parameters g_i and h of the unlearned vector coefficients $\Delta\alpha_l$.

$$\Delta g_i = \sum_{k \in S} Q_{ik}\Delta\alpha_k + \sum_{l \in U} Q_{il}\Delta\alpha_l + y_i\Delta b = 0 \; \forall i \in S \quad (4)$$

$$\Delta h = \sum_{k \in S} y_k\Delta\alpha_k + \sum_{l \in U} y_l\Delta\alpha_l = 0 \quad (5)$$

The process of perturbation is controlled by a perturbation parameter p. When the SVM solution is perturbed from its initial 'unlearned' to final 'learned' result, p varies from 0 to 1. When $p = 0$, the solution is initialized to the previous solution, before introducing new examples. During each perturbation, p is incremented by the smallest value Δp_{min} which leads to the change for examples. When $p = 1$, all unlearned vectors have reached either of the three categories, whilst both new and old data satisfies the KKT conditions.

The IL proceeds through a sequence of 'adiabatic' steps by maintaining the KKT conditions during the perturbations. The adiabatic increment $\Delta\alpha_i$ is expressed as the product of Δp and the corresponding coefficient sensitivities. Assuming that $\Delta\alpha_k = \beta_k\Delta p(k \in S)$, $\Delta\alpha_l = \lambda_l\Delta p(l \in U)$, and $\Delta b = \beta\Delta p$, we can obtain the differential KKT conditions expressed in terms of the coefficient sensitivities by

$$\gamma_i = \frac{\Delta g_i}{\Delta p} = \sum_{k \in S} Q_{ik}\beta_k + \sum_{l \in U} Q_{il}\lambda_l + y_i\beta = 0 \; \forall i \in S \quad (6)$$

$$\frac{\Delta h}{\Delta p} = \sum_{k \in S} y_k\beta_k + \sum_{l \in U} y_l\lambda_l = 0 \quad (7)$$

where λ_l is determined by a natural choice, and β_k and β are obtained by solving the system of equations. Once the coefficient sensitivities are known, we can compute the margin sensitivities γ_i for the error, reserve and unlearned vectors. The smallest Δp in applicable conditions determines the category change and perturbation step Δp_{min}. Then we can determine Δp_{min} from the possible category changes.

Once Δp_{min} is determined, we update the coefficients for the margin vectors ($\alpha_k \rightarrow \alpha_k + \beta_k\Delta p : \forall k \in S$) and the unlearned vectors ($\alpha_l \rightarrow \alpha_l + \lambda_l\Delta p : \forall l \in U$). After noting the category change, we recompute the coefficient and the margin sensitivities and determine the next perturbation. This process repeats until $p = 1$.

When no SVM solution initially exists, in other words, all of the training examples are initially unlearned and $\{\alpha_l = 0, b = 0 : \forall l \in U\}$, the margin vector coefficients allow the preservation of the equality condition when an initial SVM solution is given. The margin vectors provide the degree of freedom to counterbalance the changes in the unlearned vector coefficients. One way is to bootstrap the process by selecting one example from each class and learning an initial SVM, another is to simply proceed with the initial perturbation and disregard condition until the margin vector set is no longer empty.

After the current SVM adapt to changes in the regularization and kernel parameters, we utilize the leave-one-out (LOO) error estimation [37]. LOO divides the original data into two groups, in which each sample is used as a validation set, and the remaining samples are used as a training set. The average of the classification accuracy of the final validation set is used to evaluate the generalization performance of the SVM.

3.4.2 Region Public Model Training

After training all incremental data of a user, we employ the technique of FL to transmit the trained support vectors to

the edge regions. FL features joint learning and cooperative modeling. Each user submits the parameters of a unified model (i.e., part of the parameters are their respective training parameters, and the remaining are 0s) when building a public model. Thus, all users can assure that their models have same dimensions. It can effectively prevent user privacy leakage during data transfer [38]. Therefore, we adopt FL to optimize service cache and protect users' feature privacy. By means of the average support vector and each region's feature $X = Average(x)$ that reflects the features x of the region's users, we train a user invocation model to reflect the probability of the services in a region to be invoked by the user. A region invocation model is then trained by employing FL to extract the parameters of all the user invocation models in an edge region.

3.4.3 User Mobility Model Training

We leverage the concept of "K-Anonymity" [29] to protect users' private information, and introduce Lagrangian interpolation method to restore user mobility information based on locations of edge servers accessed by users, while effectively protecting user location privacy.

In the user mobility model training process, the longitude and latitude coordinates (i.e., the location of the server accessed by the user) need to be mapped to a set of Cartesian coordinates. The user's mobility is modeled in line segments located by the Cartesian coordinates to predict the user's movement direction. Interpolation is a common method in mathematical modeling. It is generally suitable for situations where the data is accurate or the amount of data is small. Since the locations of the segmented mobility model is certain and in tiny quantities, linear Lagrangian interpolation [39], [40] is used to train the user mobility model.

The edge server access information extracted in Sec. 3.3 contains the user set $U = \{u_1, u_2, ..., u_m\}$ and the edge servers set $ES = \{es_1, es_2, ..., es_n\}$ in each time period. Therefore, we can obtain a set of access matrices $K = (k_{it})(i \leq m, t \leq T)$, where k_{it} is the index of the edge servers es_j accessed by user u_i in the tth time period. By extracting the access matrices of all the edge servers $G_i = (k_{i0}, k_{i1}, ..., k_{iT})$ of u_i, a set of observation values $(x_{i0}, y_{i0}), (x_{i1}, y_{i1}), ..., (x_{iT}, y_{iT})$ is obtained, where $(x_{it}, y_{it})(0 \leq t \leq T)$ are the mapped Cartesian coordinates of an accessed server's location. We divide the Cartesian coordinates of the locations into groups to ensure that the abscissa of the observed coordinates in each group changes monotonously. Interpolation is performed for each group of observed coordinates $(x_{i0}, y_{i0}), ..., (x_{it}, y_{it})$, The interpolation interval $(\hat{x}_{i0}, \hat{x}_{it})$ is the range where the interpolated nodes $\hat{x}_i \in (\hat{x}_{i0}, \hat{x}_{it})$ are. The interpolation interval is calculated as follows:

$$\hat{x}_{i0} = \lfloor min\{x_i\}/step \rfloor * step \tag{8}$$

$$\hat{x}_{it} = \lceil max\{x_i\}/step \rceil * step \tag{9}$$

where *step* refers to the length of an interpolation step. There are a total of *k* interpolation points calculated by the interpolation interval and step, to simulate the value $\hat{y}_i$ at x_i.

We can calculate the interpolated predicted value by:

$$\hat{y}_i = \sum_{p=0}^{k} (y_p * \prod_{q=0, q \neq p}^{k} \frac{\hat{x}_i - x_q}{x_p - x_q}) \tag{10}$$

We then combine each group of predicted values to get the restored user's mobility information. The element es_i in the candidate edge server set A of u_i satisfies:

$$(\hat{x}_i, \hat{y}_i) \in cov(es_i) \tag{11}$$

where $cov(es_i)$ represents the coverage of edge server es_i.

In this strategy, we consider not only the historical locations (x', y') and $(\hat{x}', \hat{y}')$ of u_i, but also the new location (x, y) generated at each moment. The mobility model is dynamically adjusted by constantly comparing with the predicted result. The process of the dynamic adjustment process is described in Algorithm 1.

Algorithm 1 Dynamic Lagrange Interpolation Prediction Process

Require: The user set U, the historical locations (x', y'), the historically predicted location $(\hat{x}', \hat{y}')$, and the current location (x, y), the edge server set ES and its distribution ES_{LOC}
Ensure: The candidate edge server set A

```
for u_i ∈ U do
    Extracting the historical locations (x', y') and the current location (x, y);
    Calculating the offset o of the mobility model by comparing with (x̂', ŷ');
    while o > threshold do
        Adjusting the interpolation step to retrain the mobility model;
        Calculating the offset o of the mobility model;
    end while
    Retraining the user mobility model on the new step;
    Obtaining the user's possible location (x̂, ŷ);
    for es_j ∈ ES do
        Reading es_j in ES_LOC
        if (x̂, ŷ) ∈ cov(es_j)) then
            es_j → A
        end if
    end for
    return A ;
end for
```

3.5 Edge User Allocation

In order to optimize the overall QoS of the focused area, our solution dynamically allocates edge servers to edge users based on the cache of edge servers and the mobility of users. Since the dynamic edge user allocation problem based on edge cache is a multi-peak and multi-variable optimization problem, there is a multi-peak functional relationship between edge user allocation and the overall QoS. The Artificial Bee Colony (ABC) algorithm solves the multivariate function optimization problem by simulating the foraging behavior of the bee colony [41]. The ABC algorithm can largely avoid falling into the local optimal solution and find the global optimal solution [42], [43], [44] .

In the ABC algorithm, the honey collection system of the bee colony consists of three parts: honey sources, hired bees and non-hired bees. Hired bees only include one type of workers–collecting bees, which are responsible for contacting a certain honey source and transmitting this information in the dance area of the hive. Non-hired bees include two types of workers: scouting bees and observing bees, which are mainly responsible for mining new honey sources. Scouting bees search for new honey sources under certain field-searching rules, and transmit information in the dance zone. Meanwhile, observing bees observe honey source information and evaluate its quality to gather high-quality honey sources. The corresponding relationship between the components of the bee colony foraging model in the ABC algorithm and the QoS optimization problem is shown in Table 1.

TABLE 1: Correspondence between the components of the ABC algorithm and the QoS optimization

Bee colony foraging model	QoS optimization problem
honey sources	user-base station allocation strategy
hired bees	allocation process initialization
observing bees	policy selection process
scouting bees	realm search process
honey collection	QoS value of the allocation policy

The traditional ABC algorithm executes an optimization process based on a one-dimensional solution vector [41], reflecting the optimization performance of each parameter. We introduce the optimization process by designing *2-D* Solution Based Artificial Bee Colony Algorithm, an ABC algorithm based on a two-dimensional solution vector, and calculate the optimization performance of each allocation strategy for comparison. The goal of the cache-based dynamic edge user allocation problem is to maximize the fitness value *Q* and make the predicted user location $(\hat{x}_{u_i},\hat{y}_{u_i}) \in cov(es_j)$, where $cov(es_j) \in ES$ (i.e., edge server set). It can be formalized as follows:

$$maximize(Q = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{k}(s_{ij} * q_{ijl})) \tag{12}$$

among them, when u_i is allocated to es_j, s_{ij} = 1, otherwise s_{ij} = 0. The service quality q_{ijl} increases with the increase of historical data q'_{ijl}, and decrease with the increase of the distance between a user and an edge server $dis(u_i, es_j)$ and the capacity of the edge server $capacity(es_j)$. Its formula is as follows:

$$q_{ijl} = q'_{ijl} + \frac{\alpha}{1 + dis(u_i, es_j)} + \frac{\beta}{1 + capacity(es_j)} \tag{13}$$

where α and β are the influence coefficients of the user-server distance and the server capacity respectively.

In order to prevent it from data divergence, the sigmoid function is used to map the historical service quality data:

$$\overline{q}_{ijl} = \frac{1}{1 + e^{-\sum_{t=0}^{T} q_{ijl}}} \tag{14}$$

where T represents the optimized current time period, and the service quality is normalized to facilitate the comparison.

The steps of the ABC algorithm for the QoS optimization are as follows:

i). Initializing the honey source. The parameters of the ABC algorithm are initialized, including the number of honey sources S_N, that is, the number of collecting bees and observing bees, the number of optimization iterations *I*, and the maximum invalid number *Invalid*. The honey sources are generated by the following formula:

$$S = (s_1^T, s_2^T, ..., s_m^T) \tag{15}$$

$$S_i = (s_{i0}, s_{i1}, ..., s_{in}) \tag{16}$$

$$\exists! s_{ij} = 1 \iff es_j \in A \tag{17}$$

where s_i represents the i_{th} dimension vector of the honey source S, that is, the allocation matrix between u_i and his/her participating n edge servers in the region. An edge server es_j is randomly selected from the set of candidate edge servers A of u_i for allocation. A user can only be allocated to one server each time. Thus, $\exists! s_{ij}$ = 1. Initializing the honey source is to generate a range of random vectors for all dimensions of each honey source through the above formula, thereby generating each initial honey source of S_N.

After determining the parameters for initialization, the ABC algorithm repeats Step *ii) - iv)* in a predefined number of iterations to find the optimal solution:

ii). Collecting bees' period. The honey source is updated during collecting bees' period as follows:

$$if\ S_i = (..., s_{ij}^T, ...)\ \&\ S_k = (..., s_{kj}^T, ...) \tag{18}$$

$$then\ S'_i = (..., s_{ij-1}^T, \varphi(s_{kj}^T), s_{ij+1}^T, ...) \tag{19}$$

where S_k ($k \in 1, ..., S_N$ and $k \neq i$) represents a neighboring honey source, and $\varphi(s_{kj}^T)$ represents a random transition of the k_{th} dimension vector of the neighboring honey source S_k, that is, randomly selecting the k_{th} user to re-allocate the user with an edge server from the feasible solution. After obtaining the new honey source, the fitness values of the collected honey sources are compared to select the better one.

iii). Observing bees' period. The observing bees' period starts after the completion of the collecting bees' period. After gathering the honey sources and returning to the hive, they share the honey source information, namely the fitness value of each feasible solution. The observing bees analyze the fitness value of each honey source and calculate the relative score of each honey source by:

$$accFitness = \frac{0.9 + Q}{max(Q)} + 0.1 \tag{20}$$

Observing bees randomly choose to follow the honey sources whose relative scores are greater than a random threshold. In the collecting bees' period, we use equations 18 and 19 to update the honey sources and choose the best honey source to preserve.

iv). Scouting bees' period. If a honey source has not been updated after executing Step *ii)* and *iii)* for *invalid* times, it means no better allocation can be found. This honey source needs to be discarded and the scouting period is started.

This process reflects the self-organized negative feedback and volatility properties of the ABC algorithm [42]. In this period, the scouting bee searches for a new honey source to replace the abandoned honey source. It means, if the user allocation strategy is not updated in *invalid* times, we will abandon it and generate a new strategy.

4 EVALUATION

The experiments are conducted on a computer with the following configuration: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19GHz, 16 GB RAM and Windows 10 OS. We use MySQL 8.0 for data processing. All the experimental models are developed in Python 3.6.

4.1 Research Questions

We perform a set of dedicated experiments to explore the following basic research questions.

- RQ1: What are the optimal parameters for the user invocation model and user mobility model training?
- RQ2: How can the model training be accelerated with the IL strategy adopted in the MENIFLD_QoS method?
- RQ3: How do the parameters in the ABC algorithm, such as the iteration times and population size (i.e., the number of collecting bees), and the edge region size (i.e., the number of users) influence the optimization performance?
- RQ4: How is the performance of the proposed method in comparison to other optimization methods?

4.2 Experimental Setup

4.2.1 Data Set Description

The experimental data in this paper mainly originates from two data sets. The first part [20] [45] comes from the Shanghai Telecom data set [1], including the distribution of edge servers and users' server access information. The second part [46] originates from the open source QoS (including response time *RT* and throughput *TP*) data of different edge services. Here we use *RT* to calculate the fitness values of user allocation in an edge region to measure the optimization performance, considering the impact of server-user distance changes caused by users' mobility on the response time of an edge service.

The first data set contains more than 7.2 million records generated by 9481 users' mobile devices when they accessed to 3233 edge servers of *Shanghai* Telecom during 30 days. The edge servers of the *Shanghai* Telecom are divided into 20 regions by using the proposed AP clustering algorithm, where the edge servers in each region are marked in Fig. 7. Examples of the edge server access records are shown in Table 2. The second data set includes 142 users' QoS evaluations on 4500 services during a period of 16 hours (4 collections/hour). Examples of the QoS evaluation data are shown in Table 3.

Our experiments are conducted upon the QoS data (i.e., *RT*) of 4500 edge services accessed by 142 users from 274 edge servers within a duration of 16 hours by merging the aforementioned two data sets. We randomly select the edge server access records of 142 users' mobile devices within a duration of 16 hours to map with the QoS data set.

1. http://sguangwang.com/TelecomDataset.html

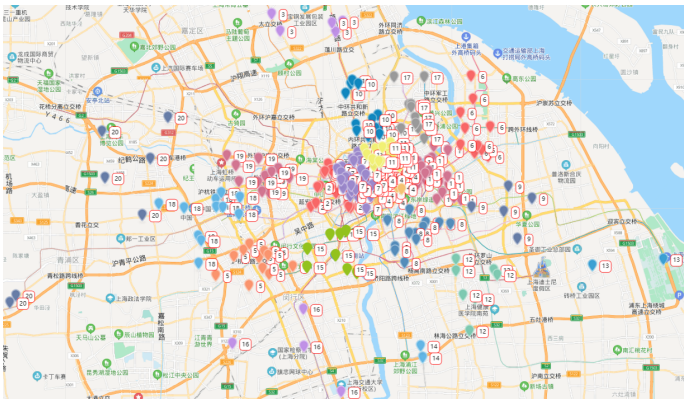


Fig. 7: Regions generated by the AP clustering algorithm

TABLE 2: *Shanghai* Telecom Data Set

Region ID	Base Station ID	Access Time	Base Station Location	User ID
5	68	22:19-22:42	31.240874/121.518086	1
1	1	17:17-17:18	31.237872/121.470259	3

4.2.2 Experimental Data Preprocessing

Cloud computing can be viewed as a centralized big data processing paradigm, while MEN can be regarded as a distributed big data processing paradigm. QoS data sets collected from cloud environments usually contain users' IP addresses and IDs and QoS values of services accessed by users as well as access time, which cannot reflect the distributed and dynamic features of the edge environment. Therefore, QoS data sets collected in clouds are required to be allocated to edge servers according to users' edge server access records. The data allocation comprises the following steps.

We assume that, in a sequence of time periods $T_i (i = 1, ..., t)$, a set of users U accesses a set of edge services WS located in a set of edge servers ES. The QoS values of the services accessed by the users together with the service access time are stored in a cloud during this process. First, we extract the users' server access records in each time period T_i from network operators, which contain the users' IP addresses and IDs and the locations of the servers accessed by the users. Next, we divide the cloud based QoS data set into t parts according to their corresponding time periods and extract the IDs and QoS data of the services accessed

TABLE 3: Time-Aware QoS Data Set

User ID	Time ID	Web Service ID	RT	TP
1	41	0	0.721	0.5034674
25	54	1012	0.309	4.579288

by each user in each time period. Finally, we map the users' server access records and QoS data of the accessed services by matching between the user and server access time of the former and the user and service access time of the latter. In this way, we can obtain the time of a service (identified by the service ID) accessed by a user (identified by the user's IP address and ID) from an edge server (identified by the server location) (termed as *service access information*) and the QoS values of the service in this process. After the edge region division, users' service access information and service QoS values are allocated to the corresponding edge region of each edge server to form an edge based QoS data set. The data set forming process is illustrated in Fig. 8.

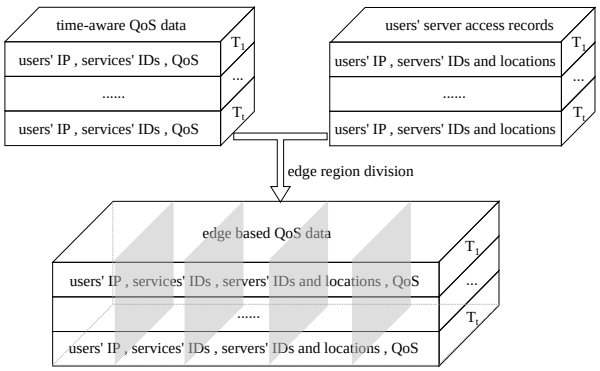


Fig. 8: Edge based QoS data set forming process

This project aims to optimize the regional QoS by optimizing the service cache and predicting the direction of user movement. Therefore, for each region, it is required to extract a user' service access information to indicate the user's service preference and the user's edge server access record to show his/her movement.

4.2.3 Experimental Parameters

The parameter setting of the experimental environment is shown in Table 4. In the user invocation model, we set the user feature as a 3-dimensional vector and generate the feature values by normal distribution. Considering that the dimension size of the samples is small, we employ Gaussian kernel as the kernel type to map the user samples to a higher dimensional space. We set the proportion parameters of distance and resource in the fitness value calculation to respectively 0.5 and 0.2 via experiments. We then configure the maximum invalid number of solutions in the ABC algorithm to 1 to promote the renewal of feasible solutions. We set the number of iterations as (0, 1, ... , 20) and the number of collecting bees S_N as (2, 4, ... , 14) respectively to find the optimal parameters. We divide the regions into four clusters according to their user scales, that is, [0, 9], [10,19], [20,29], [30,∞), and randomly choose a region from each cluster. The profile of the experimental regions are shown in Table 5. These regions can be utilized to explore how region size impacts on the optimization performance. The distribution of the edge servers in these experimental regions is shown in Fig. 9.

4.2.4 Metrics

The following experiments intuitively compare the optimization performance among the combinations of different

TABLE 4: Parameter Setting of Experimental Environment

Name	Meaning	Setting
n	number of edge servers	274
m	number of region users	142
k	number of edge services	4500
p	dimension of user feature	3
type	kernel type	Gaussian kernel with variance 1/(2*scale)
α, β	parameters in QoS calculating	0.5, 0.2
invalid	maximum invalid time in ABC	1

TABLE 5: Profile of Experimental Edge Regions

Region ID	User number	BSs number	Records number
1	49	44	2632619
4	12	16	1804082
8	24	25	2255050
10	8	15	562891

training scales, iteration times, and optimization strategies based on *fitness values* and *optimization amounts*. The *fitness value* (equation 12) indicates the overall QoS value in each edge region.

We introduce the metric of *optimization amount* to intuitively reflect the performance of the optimal solution generated by each optimization strategy in comparison to all feasible solutions. It is defined as the difference between the *fitness value* of the optimal solution and the average *fitness value* of all feasible solutions:

$$\delta = fitness_{opt} - fitness_{avg} \tag{21}$$

4.3 Experimental Procedure

After forming the edge based QoS data, we extract the data of each region to train the user mobility model and the user invocation model for each user to predict the candidate edge server to be accessed and the cache of the edge

Fig. 9: Distribution of edge servers in edge regions

region storing the services to be invoked by the user. We measure the training speed of MENIFLD_QoS based on IL in comparison to its variation based on static learning. Next, we tune the population size and the iteration number for the proposed ABC algorithm to observe their influence on the optimization effect. Eventually we compare the optimization performance of MENIFLD_QoS with several baseline strategies. The detailed experimental steps are described as follows:

(1) We compare the effect of the number of interpolation steps on the user mobility model and compare the training time on different soft-margin regularization and kernel scale parameters, to choose the appropriate parameters.
(2) We compare our proposed IL based SVM with a traditional SVM model to verify that if IL can effectively accelerate training time. To further testify the efficiency of MENIFLD_QoS for the region invocation model training, we compare its training time with another popular supervised learning based classification algorithm, the gradient descent algorithm for logistic regression.
(3) We explore how the number of iterations impacts the optimization performance of the ABC algorithm in the same edge region with the same number of collecting bees.
(4) We investigate how the population size (i.e., the number of collecting bees) influences the optimization performance of the ABC algorithm in the same edge region with the same number of iterations.
(5) We measure the effect of the edge region size on the optimization effectiveness of the algorithm based on the same number of iterations and the same number of collecting bees. We also calculate the computation time of user allocation in different edge regions.
(6) We compare the optimization performance and the number of allocated users between MENIFLD_QoS and several existing optimization methods over the experimental regions with the same population size. We also analyse their optimization trend by increasing the number of iterations.

4.4 Experimental Results

4.4.1 The Optimal Parameters for Model Training

In the user mobility model training, the number of interpolation steps influences the model's accuracy and complexity. The model predicts the set of candidate edge servers that the user may access based on locations of edge servers historically accessed by the user. We introduce the performance metric of *access rate* to measure the ratio of the number of users who access the predicted candidate edge servers to the total number of users. The *access rate* of the user mobility model at different numbers of interpolation steps in all the experimental regions is shown in Fig. 10. It can be seen that the access rate of the user mobility model decreases with the increase of the number of interpolation steps. The *access rate* values of the number of interpolation steps ranging from 100 to 500 are relatively higher. Further analysis shows that there are 31 inactive users out of 142 users in our experiments, i.e., 22% of users do not access any edge servers. Although our access rate calculation takes into account those users, the prediction error does not affect those inactive users at all. There is an additional situation where a user's actual location is within the coverage of the predicted candidate edge server but the user chooses to access another edge server. For example, when the interpolation step is 200, the mobility model accurately predicts accessed edge servers for 83 of 111 (i.e. 74.77%) active users, in addition to 12 (i.e. 10.81%) active users whose locations are within the coverage of the predicted severs.

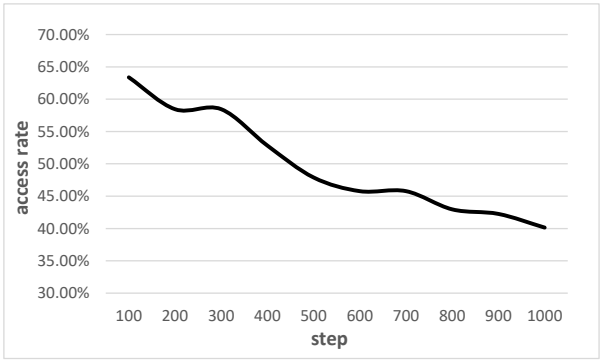


Fig. 10: *access rate* of different interpolation steps

We introduce the performance metric of *offset* to further observe the deviation between the predicted and actually accessed interpolation points. *offset* calculates the average deviation between the predicted and the actually accessed interpolation points of each user. The *offset* of the user mobility model at different numbers of interpolation steps ranging from 100 to 500 is shown in Fig. 11. We can observe that the offset is lower when the step is 200 or 400. It is acknowledged that the number of interpolation points to be computed is lower when the number of interpolation steps is larger. Therefore, we set the number of interpolation steps as 400 to balance among *access rate*, *offset* and the *computation cost*.

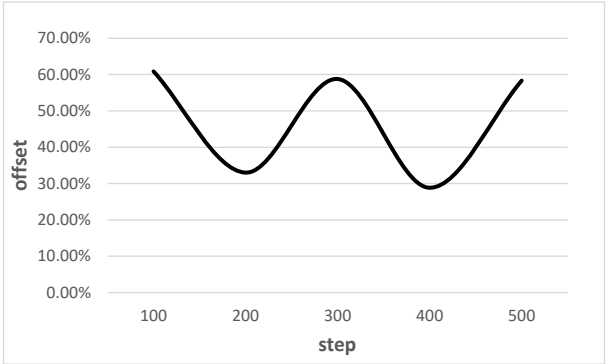


Fig. 11: *offset* of different interpolation steps

The user invocation model is built on IL and FL based SVM, the training process of which includes four steps: dynamic SVM training, leave-one-out error rate estimation (for generalization performance evaluation), regularization parameters perturbation, and kernel parameter perturbation. The training inputs include the samples X of users in a region (indicating the users' features), their service invocation labels y_k (indicating the invocation of the user sample

X on the services $service_k$), the dimension parameter of the soft-margin regularization C, and the kernel scale *scale*. We conduct a series of experiments in the four regions to explore the impact of C and *scale* on training time on average. The experimental results are shown in Fig. 12. It can be observed that there is no obvious change in the training time when C and *scale* vary. We compare the classification performance among all the possible combinations between the values of C and *scale* and find that no error vector is generated when $(C, scale) \in \{(1,6),(4,7),(3,6),(4,9)\}$. Eventually we choose $C = 1$ and *scale* = 6, since they generate lower error vectors rate and need less training time.

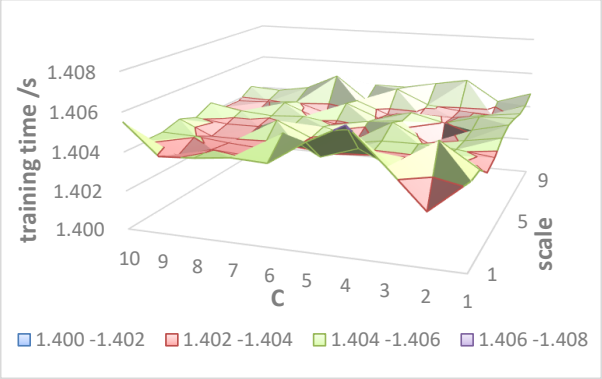


Fig. 12: Training time of soft-margin regularization parameter and kernel scale

4.4.2 Training Time

We compare the training time of three candidate service invocation models, i.e., the proposed IL based SVM model, a traditional SVM model without adoption of IL, and a logistic regression model. We make use of the three candidate models to train the region invocation model for an edge region (i.e., Region 10) randomly selected from the four experimental regions. The new data arrives in 19 of the 64 time periods. The models are trained sequentially upon the corresponding data labelled by each of the time periods. The total training time of all the 4500 services of this region in each time period is calculated. The result is shown in Fig. 13. The inference time refers to the time for estimating the possibility of invocations according to the overall preference of the region. It is relatively short (about 8s for all the users in the data set) compared with the training time. In addition, the inference time spent at each moment is almost equal. Therefore, the inference time does not need to be compared and can be ignored here.

It can be observed that the training cost of the logistic regression model is relatively stable and the highest among the three models. This is because the logistic regression model needs to be trained on all the previously arrived data when new data arrives. In contrast, the structure of SVM is more flexible and gets larger with the incremented data Therefore, its time assumption remains increasing. The IL based SVM model costs less training time than the traditional SVM, saving almost 75.8% training time of the latter. The logistic regression model is relatively simple, which is more suitable for large-scale linear classification. Its training time after time period 45 is slightly lower than

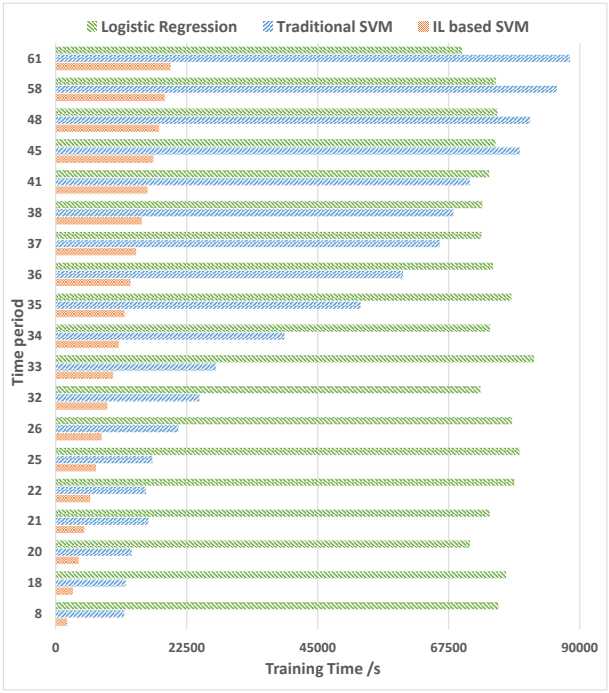


Fig. 13: Comparison of the training time

SVM, while SVM costs less training time in most of the time periods. Moreover, IL based SVM only needs to train incremented data and appropriately add disturbances to achieve the training effect in each time period, which can greatly increase the training speed.

4.4.3 Optimization Performance

Iteration Times. The average optimization performance over the experimental regions with different iteration times is revealed through the metrics of *fitness value* and *optimization amount* in Fig. 14. It can be clearly observed that the *fitness value* reaches its peak and stabilises when the number of iterations >9. On the other hand, the *optimization amount* gradually decreases with the growth of the iteration times when the number of iterations >9. This phenomenon indicates that the relative advantage of the optimal solution compared to all the feasible solutions is reduced with the higher numbers of iterations. The ideal optimization objective is to achieve a higher *fitness value* and *optimization amount* in a smaller number of iterations. Eventually we set the iteration times of the ABC algorithm to 10 by considering the balance between the *fitness value* and the *optimization amount*.

Population Size. The average *fitness value* and *optimization amount* over the experimental regions with different population sizes at the 10*th* iteration are shown in Fig. 15. The *fitness value* and *optimization amount* keep growing in a fluctuating way with the increase of the population size. The former indicates that the optimal solution can be obtained if the population size keeps expanding. The latter reveals that the larger population size can create more significant difference between the optimal and the average *fitness value*. This results from more non-optimal solutions generated by a larger population size. Compared with the population size of 2, when the population size is 4, the improvement of

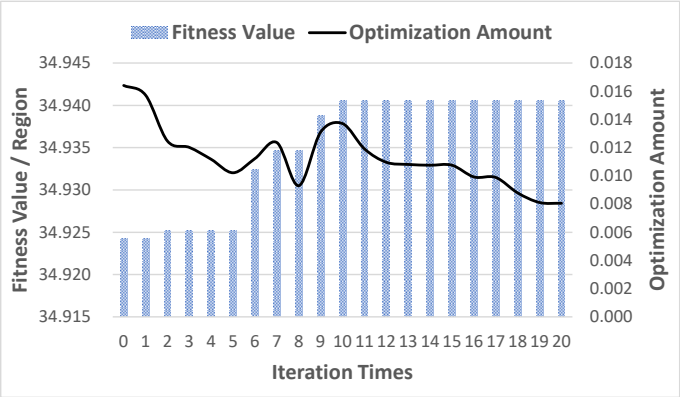


Fig. 14: The influence of iteration times on the optimization performance

the optimal fitness value is greater than the improvement of the average fitness value (i.e., $\Delta opt = fitness_{opt\_4} - fitness_{opt\_2}$, $\Delta avg = fitness_{avg\_4} - fitness_{avg\_2}$, $\Delta opt > \Delta avg$), so the *optimization amount* (equation 21, i.e., $\delta_4 = fitness_{opt\_4} - fitness_{avg\_4}$) becomes larger, and the first peak appears. Conversely, compared to the population size of 4, when the population size is 6, the improvement of the optimal fitness value is less than the improvement of the average fitness value (i.e., $\Delta opt = fitness_{opt\_6} - fitness_{opt\_4}$, $\Delta avg = fitness_{avg\_6} - fitness_{avg\_4}$, $\Delta opt < \Delta avg$), so the *optimization amount* (i.e., $\delta_6 = fitness_{opt\_6} - fitness_{avg\_6}$) becomes smaller, and the first valley appears. The *fitness value* of the solution experiences a sharp rise when the population size is 12. The *optimization amount* also reaches its peak value, indicating the greater relative advantage of the optimal solution in this point. We eventually choose 12 as the optimal population size for the ABC algorithm, considering that further expanding the population size will dramatically increase the optimization cost, and the tradeoff between the *fitness value* and the *optimization amount*.

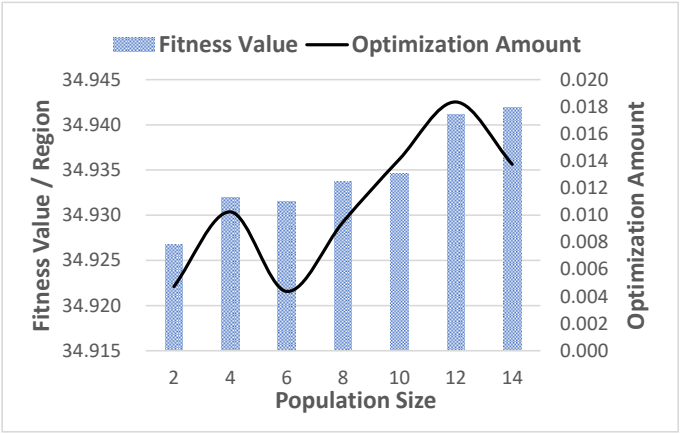


Fig. 15: The impact of population size on the optimization performance

Edge Region Size. We compared the optimization performance among ten edge regions to explore the relationship between optimization performance and the edge region size (i.e., the numbers of edge servers in a region). As shown in Fig. 16, either *optimization amount* or *fitness value* per user has no obvious trend with the increase of the region size. This indicates that the edge region size has no significant impact on the optimization performance. From equation 12, it can also be seen that the optimization effect has a non-linear relationship with the region size m.

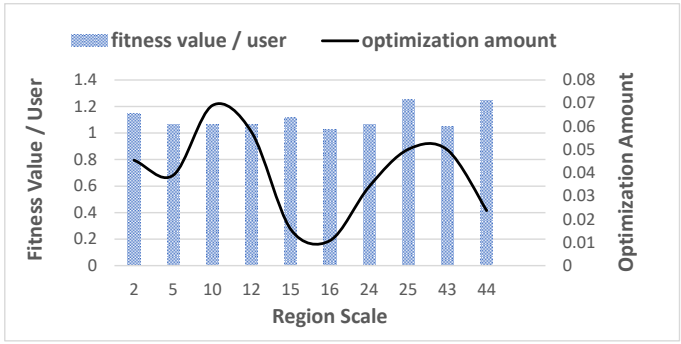


Fig. 16: The influence of edge region size on the optimization performance

Computation Time. Fig. 17 shows the computation time of user allocation in the ten edge regions. It can be seen that the growth rate of computation time is lower than it of number of users (e.g., 0.02s for 10 users, 0.13s for 89 users). The highest computation time is only 0.14s, which is relatively minor for most application scenarios (i.e., walking, cycling, driving, etc.).

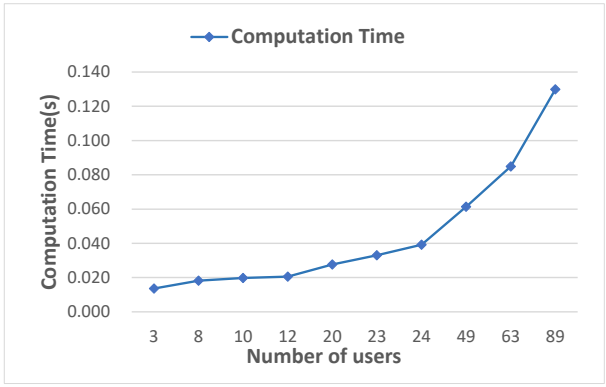


Fig. 17: The influence of number of users on the computation time

4.4.4 Optimization Method Comparison

Since the existing strategies only focus on static optimization, we compare the one-time optimization performance among our proposed strategy, two baseline strategies, and two state-of-the-art EUA strategies. We take into account users' mobility and regions' cache ahead of time to determine the solution space of each strategy. The five comparative strategies include:

- *Random*. This strategy randomly assigns edge servers to users as long as the users are in their coverage.
- *GA_QoS*. This is a variant of our proposed *MENI-FLD_QoS* strategy that bases on a genetic algorithm.
- *EUA_ILP*. This strategy employs a heuristic approach based on integer linear programming (ILP) to select

an edge server with sufficient computing resources for a user from the user's candidate server set [10] .

- *EUA_FOA*. This strategy adopts a Fruit fly Optimization Algorithm (FOA)-based approach to allocate edge users to edge servers [47].
- *MECMA_QoS*. This strategy does not train a user invocation model and does not contain FL. It only considers the mobility of user and employs the artificial bee colony algorithm to form the allocating strategy.

Fig. 18 shows the optimization performance comparison between *MENIFLD_QoS* and the other strategies. Since the purpose of the optimization is to find a better user-edge server allocation plan, the *fitness values* of the six strategies among the experimental regions are compared, along with the increased number of iterations. As can be seen, our *MENIFLD_QoS* strategy achieves the highest *fitness value* in the whole period and stays relatively stable after the second iteration, *MECMA_QoS* performs slightly weaker, followed by the *Random* and *GA_QoS* strategies whose values are more fluctuating. The lower and unstable value of *GA_QoS* results from its defect that it is easily stranded by local optimal solutions during the searching process. The *EUA_ILP* and *EUA_FOA* strategies have relatively inferior optimization performance, since they cannot guarantee every user in a region to be allocated to an edge server. *EUA_ILP* is a heuristic method and its value tends to be flat after the third iteration. *EUA_FOA* is based on a swarm intelligence technique and achieves higher values than *EUA_ILP*. However, it converges slowly and also easily falls into local optimal solutions during the searching process.

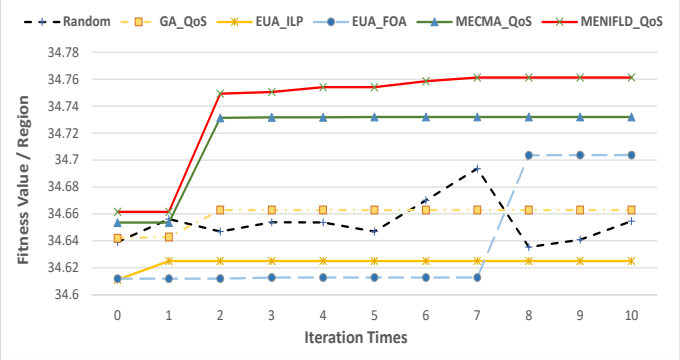


Fig. 18: Comparison of the optimization performance

Next, we compare the numbers of successfully allocated users in our four experimental regions among those strategies in Fig. 19. It can be seen that *Random*, *GA_QoS*, *MECMA_QoS* and *MENIFLD_QoS* can allocate all the users in the experimental regions, in comparison to the two state-of-the-art EUA strategies, *EUA_ILP* and *EUA_FOA*. This is because *Random*, *GA_QoS*, *MECMA_QoS* and *MENIFLD_QoS* have the same objective function that aims to allocate all the users to edge servers in an edge region, whilst *EUA_ILP* and *EUA_FOA*'s objective functions cannot guarantee all the users to be allocated to edge servers in an edge region.

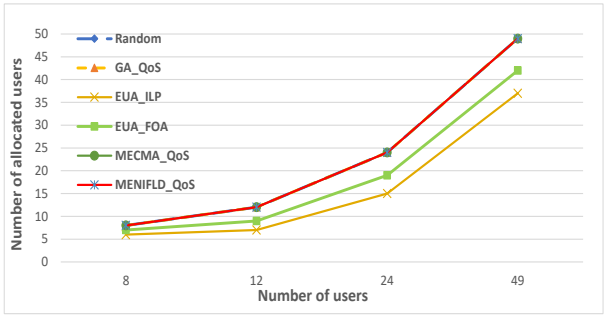


Fig. 19: Number of allocated users

5 CONCLUSIONS AND FUTURE WORK

The existing QoS optimization strategies do not consider the allocation of edge users, both the user feature privacy and location privacy, and the impact of dynamic incremental data. Further, their research scenarios are prone to single and static. In response to the above problems, this paper proposes a mobile QoS optimization strategy based on IL and FL in the network. By optimizing the edge service cache and user mobility scenarios, the user's feature privacy and location privacy are effectively protected. An improved artificial bee colony algorithm basing on the two-dimensional solution to the edge network is proposed. As a result, user allocation problem is optimized.

The experiment proves that the edge cache can optimize the regional QoS. The improved artificial bee colony algorithm effectively optimizes the user allocation according to its multi-variable and multi-peak characteristics. The idea of IL can significantly improve the efficiency of model training. Therefore, the MENIFLD_QoS strategy proposed in this paper effectively realizes privacy protection while ensuring the optimization performance, and effectively enhances the cache model training efficiency.

In future work, we will focus on the following issues. First, at present, only response time is considered to optimize regional QoS, we will further study multiple QoS attribute values for QoS optimization. Second, the current approach is only locally optimized in a clustering region, we will consider QoS and load balancing for global optimization.

ACKNOWLEDGMENTS

This work is funded by the National Natural Science Foundation of China under Grant No.62272145 and No.U21B2016, the 2021 Postgraduate Research and Practice Innovation Program of Jiangsu Province under grant number KYCX21_0549, and the Fundamental Research Funds for the Central Universities under grant number B210203070 and B210202075.

This research was also supported partially by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP220101823). The views expressed herein are those of the authors and are not necessarily those of the Australian Government or Australian Research Council.

REFERENCES

[1] P. Zhang, H. Jin, H. Dong, W. Song, and A. Bouguettaya, "Privacy-Preserving QoS forecasting in mobile edge environments," *IEEE Transactions on Services Computing,*, vol. 15, no. 2, pp. 1103–1117, 2022.

[2] W. Shi, H. Sun, J. Cao, Q. Zhang, and W. Liu, "Edge computing-an emerging computing model for the internet of everything era," *Computer Research and Development*, vol. 54, pp. 907–924, 2017.

[3] R. M. Sreenath and M. P. Singh, "Agent-based service selection," *Web Semantics Science Services Agents on the World Wide Web*, vol. 1, no. 3, pp. 261–279, 2004.

[4] A. Gohil, H. Modi, and S. K. Patel, "5g technology of mobile communication: A survey," in *2013 international conference on intelligent systems and signal processing (ISSP)*, pp. 288–292, IEEE, 2013.

[5] F. Han, S. Zhao, L. Zhang, and J. Wu, "Survey of strategies for switching off base stations in heterogeneous networks for greener 5g systems," *IEEE Access*, vol. 4, pp. 4959–4973, 2016.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *Internet of Things Journal, IEEE*, vol. 3, no. 5, pp. 637–646, 2016.

[7] T. Mohammed, A. Albeshri, I. Katib, and R. Mehmood, "Ubipriseq—deep reinforcement learning to manage privacy, security, energy, and QoS in 5g iot hetnets," *Applied Sciences*, vol. 10, no. 20, p. 7120, 2020.

[8] Josh Taylor, "Optus reveals at least 2.1 million ID numbers exposed in massive data breach." https://www.theguardian.com/business/2022/oct/03/optus-commissions-independent-review-of-data-breach, 2022.

[9] John Davidson, "Optus breach was huge, but companies lose our data all the time." https://www.afr.com/technology/optus-breach-was-huge-but-companies-lose-our-data-all-the-time-20220930-p5bmcs, 2022.

[10] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *International Conference on Service-Oriented Computing*, pp. 86–101, Springer, 2019.

[11] J. Xu, X. Li, X. Liu, C. Zhang, L. Fan, L. Gong, and J. Li, "Mobility-aware workflow offloading and scheduling strategy for mobile edge computing," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 184–199, Springer, 2020.

[12] Y. Hu, H. Shen, G. Bai, and T. Wang, "Privacy-preserving task allocation for edge computing enhanced mobile crowdsensing," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 431–446, Springer, 2018.

[13] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Generation Computer Systems*, vol. 85, pp. 1–8, 2018.

[14] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[15] Y. Miao, G. Wu, M. Li, A. Ghoneim, and M. S. Hossain, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Generation Computer Systems*, vol. 102, pp. 925–931, 2020.

[16] S. Deng, Z. Xiang, J. Taheri, M. A. Khoshkholghi, J. Yin, A. Y. Zomaya, and S. Dustdar, "Optimal application deployment in resource constrained distributed edges," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1907–1923, 2020.

[17] Y. Qian, L. Hu, J. Chen, X. Guan, and A. Alelaiwi, "Privacy-aware service placement for mobile edge computing via federated learning," *Information Sciences*, vol. 505, pp. 562–570, 2019.

[18] C. Wu, Q. Peng, Y. Xia, and J. Lee, "Mobility-aware tasks offloading in mobile edge computing environment," in *2019 Seventh International Symposium on Computing and Networking (CANDAR)*, pp. 204–210, IEEE, 2019.

[19] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[20] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.

[21] V. Oleshchuk, "Internet of things and privacy preserving technologies," in *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*, pp. 336–340, IEEE, 2009.

[22] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv preprint arXiv:1602.05629*, vol. 2, 2016.

[23] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *International Conference on Service-Oriented Computing*, pp. 230–245, Springer, 2018.

[24] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, S. Pang, H. Liu, Y. Qin, *et al.*, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *2019 IEEE International Conference on Web Services (ICWS)*, pp. 91–98, IEEE, 2019.

[25] Y. Liang, J. Ge, S. Zhang, J. Wu, L. Pan, T. Zhang, and B. Luo, "Interaction-oriented service entity placement in edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1064–1075, 2019.

[26] K. You, B. Tang, Z. Qian, S. Lu, and D. Chen, "Qos-aware placement of stream processing service," *Journal of Supercomputing*, vol. 64, no. 3, pp. 919–941, 2013.

[27] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[28] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, E. Bertino, and K. Y. Lam, "Privacy preserving location-aware personalized web service recommendations," *IEEE Transactions on Services Computing*, vol. 14, no. 3, pp. 791–804, 2018.

[29] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[30] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Encrypted database integrity in database service provider model," in *IFIP World Computer Congress, TC 11*, pp. 165–174, Springer, 2002.

[31] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[32] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[33] E. Grafarend, "The optimal universal transverse mercator projection," in *Geodetic Theory Today*, pp. 51–51, Springer, 1995.

[34] Y. Xiao and J. Yu, "Semi-supervised clustering based on affinity propagation algorithm," *Journal of software*, vol. 19, no. 11, pp. 2803–2813, 2008.

[35] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[36] C. P. Diehl and G. Cauwenberghs, "Svm incremental learning, adaptation and optimization," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, pp. 2685–2690, IEEE, 2003.

[37] O. Chapelle and V. Vapnik, "Model selection for support vector machines," *Advances in neural information processing systems*, vol. 12, pp. 231–236, 1999.

[38] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[39] B. Li, H. Zhang, and H. Lu, "User mobility prediction based on lagrange's interpolation in ultra-dense networks," in *2016 IEEE 27th annual international symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2016.

[40] T. Sauer and Y. Xu, "On multivariate lagrange interpolation," *Mathematics of Computation*, vol. 64, no. 211, pp. 1147–1170, 1995.

[41] X. Wang, Z. Wang, and X. Xu, "An improved artificial bee colony approach to qos-aware service selection," in *2013 IEEE 20th International Conference on Web Services*, pp. 395–402, IEEE, 2013.

[42] E. Bonabeau, M. Dorigo, G. Theraulaz, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. No. 1, Oxford university press, 1999.

[43] S.-I. Bejinariu, H. Costin, F. Rotaru, R. Luca, and C. D. Niţă, "Performance analysis of artificial bee colony optimization algorithm," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, pp. 1–4, IEEE, 2017.

[44] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied mathematics and computation*, vol. 214, no. 1, pp. 108–132, 2009.

[45] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "Qos prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 134–144, 2019.
[46] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2012.
[47] T. Li, W. Niu, and C. Ji, "Edge user allocation by FOA in edge computing environment," *Journal of Computational Science*, vol. 53, p. 101390, 2021.

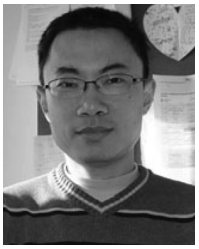
Huiying Jin is a PHD candidate with the College of Computer and Information, Hohai University, Nanjing, China. She received her bachelor degree in Software Engineering from Yangzhou University, Yangzhou, China in 2017. Her current research interests include services computing and data mining. She has won National scholarship for doctoral students. She has published in international journals such as *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Services Computing* and *Information and Software Technology*.

Pengcheng Zhang received the Ph.D. degree in computer science from Southeast University in 2010. He is currently a full professor in College of Computer and Information, Hohai University, Nanjing, China. His research interests include software engineering, service computing and data science. He has published research papers in premiere or famous computer science journals, such as *IEEE Transactions on Big Data*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Emerging Topics in Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Reliability*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Software Engineering*, *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Data Science*. He was the co-chair of IEEE AI Testing 2019 conference. He served as a technical program committee member on various international conferences. He is a member of the IEEE.

Hai Dong received a PhD from Curtin University. He is currently a Senior Lecturer at School of Computing Technologies in RMIT University, Melbourne, Australia. He was previously a Vice-Chancellor's Research Fellow in RMIT University and a Curtin Research Fellow in Curtin University, Perth, Australia. His primary research interests include: Service-Oriented Computing, Edge Computing, Blockchain, Cyber Security, Machine Learning and Data Science. His publications appear in *ACM Computing Surveys*, *IEEE Transactions on Industrial Electronics*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Software Engineering*, etc. He currently chairs the IEEE Computational Intelligence Society Task Force on Deep Edge Intelligence. He is a Senior Member of the IEEE.

Xinmiao Wei is a master student in the College of Computer and Information, Hohai University, Nanjing, China. She received her bachelor degree in Software Engineering from Yangzhou University, Yangzhou, China in 2019. Her current research interests include service computing and data mining. She has won National scholarship for postgraduates.

Yuelong Zhu is currently a Professor and a Ph.D. Supervisor with the College of Computer and Information, Hohai University, Nanjing, China. He is the Deputy Chairman of the Water Resources Informatization Special Committee of the China Hydraulic Engineering Society, and the Vice Chairman of the Jiangsu Water Resources Protection Association. His main research interests include intelligent information processing and data mining, and water conservancy informatization. He was awarded the National Second Prize for Scientific and Technological Progress, the First Prize for Excellent Achievements of the Ministry of Education, and the Dayu Water Science and Technology First Prize.



Tao Gu is currently a Professor in Department of Computing at Macquarie University, Sydney. He obtained his Ph.D. in Computer Science from National University of Singapore, M.Sc. in Electrical and Electronic Engineering from Nanyang Technological University, and B.Eng. in Automatic Control from Huazhong University of Science and Technology. His current research interests include Internet of Things, Ubiquitous Computing, Mobile Computing, Embedded AI, Wireless Sensor Networks, and Big Data Analytics. He is a senior member of the IEEE.