Profit-aware Edge Server Placement

Yuanzhe Li, Ao Zhou, *Member, IEEE*, Xiao Ma, *Member, IEEE* and Shangguang Wang, *Senior Member, IEEE*

***Abstract*—In 5G network, mobile edge computing plays a key role in providing low access delay services. The placement of edge servers not only determines the quality of services on the user side, but also affects the profit of running a mobile edge computing system. In this paper, we study how to properly place edge servers so as to guarantee the access delay and maximize the profit of edge providers. We first propose a profit model which involves both access delay and energy consumption. In this model, we take 5G User Plane Function (UPF) into consideration to calculate access delay for the first time. Then we devise a particle swarm optimization based algorithm to optimize the profit. In the algorithm, we introduce a weight value q to guarantee the access delay and assign base stations properly. Moreover, a service level agreement is adopted to balance the trade-off between access delay and energy consumption. We take advantage of our 5G network emulator called mini5Gedge and dataset from Shanghai Telecom to conduct massive experiments. The results show that our algorithm stands out in terms of achieving the highest profit.**

***Index Terms*—5G, Mobile Edge Computing, User Plane Function.**

I. Introduction

A. Background & Motivation

MOBILE Edge Computing (MEC) is emerging as a key enabler of 5G. By sinking computation resources to the edge of the network, the long latency resulting from long distance between user device and cloud data center is mitigated. A typical mobile edge computing system is shown in Fig. 1. In this system, edge servers are deployed at base stations to form edge nodes. Base stations are assigned to edge nodes in their proximity and forwarding service requests to edge nodes. Applications benefit from the resources provided by edge nodes to guarantee a high quality of service.

Deploying such a system to achieve the full promise of mobile edge computing is quite a challenging work. On the one hand, services running in edge servers, such as VR/AR and Internet of Vehicles, are usually delay-sensitive and computation-intensive. This requires edge servers to be as close to users as possible. On the other hand, resources per edge node is quite limited compared to the large data centers. This means one edge node cannot cover a large area. Taking all the above factors into consideration, deploying more edge nodes becomes an inevitable choice to guarantee the quality of services and prevent coverage holes [1] [2] in the city.

However, edge server placement is not simply choosing locations for edge servers. It involves deploying computation resources in the city and distributing these resources to users.

Yuanzhe Li, Ao Zhou, Xiao Ma and Shangguang Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.
E-mail:{buptlyz;aozhou;maxiao18;sgwang}@bupt.edu.cn

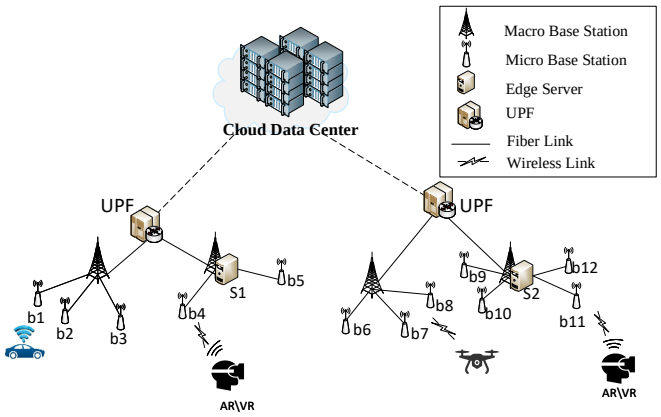

Fig. 1. A mobile edge computing system.

Where to place edge server and how to assign user equipments both matter in this process. A good placement scheme should take multiple factors into consideration. First, the access delay of user equipments should be guaranteed. Long access delay will deteriorate quality of experience for applications such as AR/VR. For Internet of Vehicles, long access delay can even result in traffic accidents. Second, the energy consumption of edge servers should be considered. As computation infrastructures, edge servers are energy intensive [3]. In a poor placement scheme, large number of servers working in idle state will cause unnecessary energy waste. Third, edge server overloading should be prevented. Areas with high user density should place more edge servers to prevent service failure resulting from edge servers getting overloaded.

Note that, when calculating delay of user equipments, the architecture of mobile network should be considered. Generally, user data packets are encapsulated using General Packet Radio Service (GPRS) Tunneling Protocol and transferred to the core network in a GPRS Tunnelling Protocol for User Plane (GTP-U) tunnel [4] [5]. Then the decapsulation is conducted at packet data network gateway. After that, user data packets can enter the Internet. In previous generations of mobile communication, packet data network gateway locates at the far away core network. Even if a user stand nearby the edge server, his data traffic have to travel a long distance to reach packet data network gateway and then go back to the edge server. 5G network adopts a thorough control and user plane separation and introduces UPF [6]. Being deployed at the network edge, UPFs conduct packet processing and traffic steering at the proximity of users. As a result, the bandwidth efficiency is improved and the network congestion can be mitigated. Besides, UPFs also serve as interconnection point between mobile network and data network, protocol data unit session anchor for mobility management and uplink

classifier for packet routing [6] [7]. All these functionalities are indispensable for service offloading in mobile edge computing. That is, UPF is the inevitable node in the path of an offloading traffic. If UPFs are not properly deployed, user equipments will suffer from extra access delay. Therefore, the location of UPF should be considered in edge server placement.

From the perspective of edge providers who deploy and run an edge system, profitability is the first priority. The technical keys to make the system profitable includes two parts: guaranteeing the access delay and reducing the energy consumption. Both are important and should be jointly considered. Pursuing extreme short delay will result in large scale edge server placement, which brings tremendous cost as a result of extra energy consumption. On the other hand, deploying too few edge servers to reduce power consumption will inevitably result in excessive access delay. As access delay is the key performance indicator of an edge system, this will lead to the loss of users. Therefore, the trade-off between reducing access delay and cutting energy consumption should be balanced in order to make a high profit.

In our previous work[1], we study the energy saving of edge servers in MEC [8]. However, as mentioned above, only reducing energy consumption is not enough when it comes to maximizing profit. Besides, it is also imprecise to denote access delay as Euclidean Distance. To address these weaknesses, in this paper, a Service Level Agreement (SLA) model is added to balance the trade-off between access delay and total energy consumption. Furthermore, network topology together with the influence of UPF on access delay are considered. Note that we only study the initial placement. Edge server placement is completely different from service deployments. Once placed, the location of edge server hardly changes. Due to the high costs, dynamic deployment is not applicable in edge server placement. The modification of placement scheme mainly involves placing new edge servers for system expansion, which is not within the scope of study.

B. Technical Challenges and Solutions

The first challenge lies in the joint consideration of network topology and workload distribution. Edge server placement is conducted according to the density of user requirements. However, as mentioned above, access delay is not simply determined by the distance between edge server and user equipments. Instead, the location of UPF determines the path of user traffic. Therefore, when calculating the access delay, the relative location between the edge server and the UPF needs to be considered. Adding that edge server should provide enough computation resource for users in its coverage area, solving the problem becomes more complicated. In order to study the access delay with consideration of UPF, we build mini5Gedge, an emulator that implemented functions of 5G user plane. With mini5Gedge, the access delay in different situations can be studied.

[1]This paper is an extension to a conference paper [8]. The earlier version of this paper was presented at IEEE International Conference on Edge Computing and was published in its Proceedings(DOI: 10.1109/EDGE.2018.00016).

The second challenge is the scale of the problem. The placement of edge servers is to choose potential locations from thousands of base stations and assign base stations to edge nodes. Such a large quantity of base stations means that the search space for the problem will be enormous. For example, the total number of possible solutions of choosing 20 placement locations from 1,000 base stations is more than 3.3×10^{41}. The search space is so huge that it is quite difficult to find the optimal solution. To solve the problem effectively, we use Particle Swarm Optimization(PSO) algorithm to solve the problem. We introduce a parameter q to assign base stations more properly. To distinguish it from the original PSO, we refer to it as QPSO in this paper.

The third challenge is how to balance the trade-off between access delay and energy consumption. Ideally, the edge servers should be deployed as many as possible in order to achieve short access delay. However, deploying too many edge servers will result in huge energy consumption and a waste of hardware resources. On the other hand, too few edge servers cannot guarantee that the access delay is always good enough in different places. In cloud data centers, idle servers can be turned off to reduce energy consumption. However, as base stations are faced with limited space to deploy devices, only one edge server is placed at base station. Turning off the server will result in service coverage hole in the area. Therefore, the appropriate method is to optimize the placement scheme of edge servers. To this end, in the initialization phase, we use maximum delay, denoted as $d_{\max}$, as a constraint and introduce a weight value q to help assign base stations properly. Then during the evolution of the algorithm, we use piece-wise SLA to push algorithm to optimizing access delay in an appropriate level.

C. Limitations of Prior Art

The key limitations of existing work concerning edge server placement lie in two aspects: First, the impact of UPF is neglected. Most previous work focus on optimizing access delay [9]–[11] or total cost [12]–[14]. However, they do not take 5G network topology into consideration. UPF is the key user plane function in 5G and all user traffics get into data network through it. Placing edge servers without considering UPF and 5G network topology is impractical. Second, the energy saving of edge servers is neglect. When it comes to saving energy in MEC, most work focuses on reducing energy consumption of user devices [15]–[19]. Server energy consumption is neglected. As task offloading is the process of transferring energy consumption from mobile devices to edge nodes, energy consumption is concentrated to edge nodes. How to reduce energy consumption of edge nodes as well as guarantee the quality of service is of great importance. Although saving energy for servers is extensively studied in cloud data centers [20]–[23]. These work cannot be directly applied to MEC. Because these approaches are irrelevant to geographical distribution and are not constrained by short access delay. While edge servers are placed at different locations and faced with strict delay constraint.

D. Contributions

This work studies the problem of edge server placement in MEC from the perspective of edge providers. The main contributions are as follows:

1) We have investigated the edge server placement problem in 5G scenario. As far as we know, we are the first to take the location of UPF in the 5G network topology into consideration in edge server placement.
2) We have analyzed the importance of reducing access delay and total energy consumption in edge server placement problem and formulated it as a profit optimization problem. We have introduced a piece-wise SLA in our model to help balance the trade-off and a PSO based algorithm has been proposed to solve the problem.
3) To study the influence of UPF and calculating the access delay, we have built an emulator system which implemented the user plane of 5G network with a complete GTP-U protocol stack. In this system, user traffics are guided by UPFs by means of building different GTP-U tunnel between base stations and edge servers.
4) We have conducted experiments based on Shanghai Telecom base station dataset and evaluated the performance of our algorithm comparing with benchmark algorithms. Our approach stands out in terms of achieving the highest profit.

II. Related Work

Edge placement problem (also called cloudlet placement) is attracting more and more attention. Alejandro Santoyo-Gonzalez and Cristina Cervello-Pastor [24] point out that the placement scheme has a great effect on edge resource efficiency. They propose a list of parameters to evaluate the placement of edge servers in the emerging 5G scenario. Qiang Fan et al. [25] minimize both cloudlet cost and end to end delay cost with a Lagrangian heuristic algorithm. Feng Zeng et al. [26] focus on minimizing total number of edge servers. They use simulated annealing based approach and a greedy based algorithm to solve the problem. Song Yang et al. [27] point that placing cloudlet at each base station is energy costly and optimize the placement scheme of cloudlets to reduce total energy consumption while satisfying delay requirements. Manasvi G. et al. [28] leverage social network information to place edge servers so that the bandwidth pressure can be relieved and the access delay can be reduced. Guangming Cui et al. [29] take network robustness of distributed MEC environment into consideration. They propose an integer programming based approach to place edge servers and improve the experience of users.

The aforementioned work is effective, but can still be improved. Although most of these work takes access delay into consideration, either as the optimization objective or as a constraint, they neglect the influence of UPF on access delay. This will weaken the performance of algorithms in real-world scenarios.

Most existing work on MEC mainly focuses on saving energy for mobile devices by offloading tasks to edge servers [15] [16] [17] [18] [19] [30]. However, reducing energy consumption in edge infrastructures is of great importance, too. Although little work in edge computing focus on energy consumption of edge servers, saving energy has been studied a lot in cloud computing. For example, Yi Wang et al. [20] use mixed integer programming based algorithms to solve large-scale virtual machine(VM) placement problems. They formulate a non-linear power consumption model to obtain physical machine energy consumption. Amandeep Kaur et al. [21] try to reduce the active servers and unnecessary migrations by proposing an algorithm based on Modified Best Fit Decreasing Algorithm [22]. Most of the research mainly focused on the efficiency of servers, but Hong Yao et al. [23] concern on both the flow routing and VM placement. By deploying VMs that frequently communicate with each other on the same server, the energy consumption of data center network could be reduced. They describe the problem by adopting an Integer Liner Programming model and presented a heuristic algorithm to solve it.

The researches above are effective and most of them formulate the VM placement as a bin packing problem [31]. However, the placement of virtual machines in the same data center mainly affects the utilization efficiency of hardware resources but has little impact on network latency. While in edge computing, access delay should be taken into consideration. The geographical distribution also brings big challenges.

PSO [32] is widely used to solve optimization problem. For example, Haitao Yuan et al. [33] use hybrid simulated annealing PSO to minimize cost in cloud task scheduling. Luan N. T. Huynh et al. [34] leverage the convergence and high solution quality to computation offloading in heterogeneous MEC networks. In [35], PSO is combined with genetic algorithm to minimize total energy consumption of user devices and edge servers. Besides, a novel self-organizing PSO [36] is used to solve multiple objective optimization in 5G wireless network. Some work focuses on improving the performance of PSO. Zhiming Lv et al. [37] propose a surrogate-assisted PSO to speed up the convergence. Yulian Cao et al. [38] put forward a comprehensive learning PSO embedded with local search to improve the global search capability and fast convergence ability.

III. System Model and Problem Definitions

A. System Model

In mobile edge computing (shown in Fig. 1), the network is a three-tier architecture: cloud data center tier, edge tier and user device tier. We focus on the edge tier, which consists of n base stations. Among these base stations, there are m base stations that will be chosen as the location to place edge servers. They will act as edge nodes to provide computing and storage resources to users in the proximity. Because edge servers are placed at base stations where it is not suitable to deploy computing clusters due to the limited space, each edge node will place one edge server.

As is shown in Fig. 2, base stations and edge nodes are geographically distributed in cities. Each edge node has a serving area, which is represented by blocks in different colors. User requests in this area are sent to this edge server. The

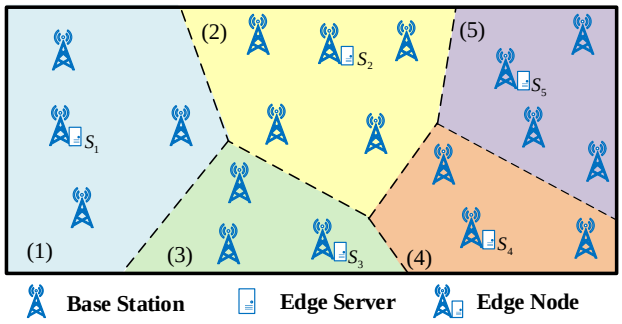

Fig. 2. Distribution of base stations and edge nodes.

network topology of edge tier can be considered as an undirected graph $G = (B \cup S, A)$, where $B = \{b_1, \ldots, b_i, \ldots, b_n\}$ denotes the set of base stations. $b_i (i = 1, 2, 3, \ldots, n)$ denotes base station i. $S = \{s_1, \ldots, s_j, \ldots, s_m\}$ denotes the set of edge servers. $s_j (j = 1, 2, 3, \ldots, m)$ denotes edge server j. Note that edge servers are all placed at base stations and every base station is a potential location to place edge server. $A = \{(b_i, s_j) | a_{b_i}(s_j) = 1, \forall b_i \in B, \forall s_j \in S\}$ represents the set of links between base stations and edge servers, where $a_{b_i}(s_j) \in \{0, 1\}$ denotes whether base station b_i is assigned to edge server s_j. Let $L = \{(s_j, b_i) | l_{s_j}(b_i) = 1, \forall b_i \in B, \forall s_j \in S\}$ denotes the set of edge server locations, where $l_{s_j}(b_i) \in \{0, 1\}$ denotes whether edge server s_j is placed at the location of base station b_i. A complete edge server placement scheme consists of edge server location selection and base station assignment. Therefore, an edge server placement scheme can be denoted as $\Omega = (L, A)$.

B. Access Delay

In edge server placement, the location of edge server and the assignment relationship mainly affect backhaul delay between edge nodes and base stations. Therefore, in this study, the delay refers to the backhaul delay. It is defined as the round-trip time between two nodes. It is denoted as $d(\beta_i, \beta_j)$, where β_i, β_j denote the two ends of a communication link, respectively. It could be a base station, an edge node or a UPF. More specifically, when it comes to the access delay between one base station and one edge server, it consists of two parts: the delay from base station to UPF and the delay from UPF to edge server. Therefore, the access delay of a user request offloaded to an edge server is defined as follows:

$$d(s_j, b_i) = min(d(s_j, u_k) + d(u_k, b_i)) \quad (\forall k, 1 \leq k \leq r), \tag{1}$$

where u_k denotes UPF $k (k = 1, 2, 3, \ldots)$ and r is the total number of UPF. That is, the access delay of a user request equals to the access delay to the edge node via the UPF with the lowest access delay.

C. Energy Consumption

There are many factors that affect the energy consumption of a server, such as the state of CPU, memory, hard disk, network card and so on. Among these factors, CPU is the most important energy-consuming device [39]. Besides, according to [40] [41], the power consumption of a server can be accurately described by a liner relationship between the power consumption and the CPU utilization efficiency. Therefore, the CPU utilization efficiency is usually used to represent the utilization efficiency of server resources [42] [43], and we can indirectly obtain the server's energy consumption according to the CPU utilization efficiency.

There is an important fact that cannot be neglected: the basic energy consumption of a server in idle state account for more than 60% of the energy consumption of a server running in full state [44] [39]. This means a great deal of energy is wasted when servers work in low efficiency or idle state. Therefore, to reduce energy consumption in edge computing, the number of servers working in low efficiency should be reduced. To this end, the edge server placement scheme should be optimized so that the efficiency of edge servers could be improved.

Given the facts above, we can model the server energy consumption as follows:

$$E_j = \int_{t_1}^{t_2} P_j(t) dt, \tag{2}$$

$$P_j(t) = P_{\text{idle}} + (P_{\max} - P_{\text{idle}}) \times \mu_j(t), \tag{3}$$

$$\mu_j(t) = \frac{w_{s_j}(t)}{w_{\max}}, \tag{4}$$

TABLE I
NOTATION TABLE

Notation	Definition/Description
n	Total number of base stations, $(n = 1, 2, 3, \ldots)$
m	Total number of edge servers, $(m = 1, 2, 3, \ldots)$
r	Total number of UPF, $(r = 1, 2, 3, \ldots)$
b_i	Base station i $(i = 1, 2, 3, \ldots, n)$
s_j	Edge server j $(j = 1, 2, 3, \ldots, m)$
u_k	UPF k $(k = 1, 2, 3, \ldots, r)$
B	The set of base stations. $B = \{b_1, \ldots, b_i, \ldots, b_n\}$
S	The set of edge servers. $S = \{s_1, \ldots, s_j, \ldots, s_m\}$
$w_{b_i}(t)$	Workload of b_i at time t
$w_{s_j}(t)$	Workload of s_j at time t
$w_{\max}$	The maximum workload of edge server
$l_{s_j}(b_i)$	Whether edge server s_j is placed at the location of base station b_i
$a_{b_i}(s_j)$	Whether base station b_i is assigned to edge server s_j
$d(\beta_i, \beta_j)$	The delay between β_i and β_j
$d_{\max}$	The maximum acceptable access delay
d_{ideal}	The maximum ideal access delay
$\mathcal{P}$	Total profit of one edge server placement scheme
N	Population size of particle swarm in the algorithm
$\mathcal{T}$	The maximum iteration number of our algorithm

$$w_{s_j}(t) = \sum_{i=1}^{n} a_{b_i}(s_j) w_{b_i}(t). \tag{5}$$

where Eq. 2 denotes the energy consumption of s_j during the time period from t_1 to t_2 ($t_1 < t_2$). E_j denotes the energy consumption of edge server s_j, $P_j(t)$ denotes the power of s_j at time t. P_{idle} denotes the power when a server works in idle state, $P_{\max}$ is the power of a server which is working in full state. $\mu_j(t)$ denotes the utilization efficiency of s_j. The value of $\mu_j(t)$ can be calculated from Eq. 4 where $w_{s_j}(t)$ denotes the workload of s_j at time t and $w_{\max}$ represents the upper limit of an edge server's work load. Then, the total energy consumption of all the edge servers can be calculated as follows:

$$E_{total} = \sum_{j=1}^{m} E_j. \tag{6}$$

Accordingly, the total cost of an edge provider brought by energy consumption is defined as follows:

$$m_{\text{cost}} = p' \times E_{total}, \tag{7}$$

where m_{cost} denotes the total cost of energy and p' denotes the price of electricity.

D. SLA Model

SLA establishes an agreement between service providers and its users. The terms in an SLA must be fulfilled when service providers provide services. In our model, we adopt an SLA with linear punishment [45]. Two parameters are introduced, i.e. d_{ideal} and $d_{\max}$ ($d_{\text{ideal}} < d_{\max}$). When the access delay is less than d_{ideal}, the quality of service is good. Edge service provider will get a full payment. However, if access delay is larger than d_{ideal}, the quality of service starts to deteriorate. As a result, the payment will reduce linearly as a punishment. If access delay exceeds $d_{\max}$, the service quality is totally unacceptable and the payment is reduced to zero. The value of d_{ideal} and $d_{\max}$ are determined by the requirement of quality of service. Accordingly, the payment in SLA can be formulated as follows:

$$p\left(s_j, b_i\right) = \begin{cases} p_{\max} & d\left(s_j, b_i\right) \le d_{\text{ideal}} \\ p_{\max} - \gamma\left(d\left(s_j, b_i\right) - d_{\text{ideal}}\right) & d_{\text{ideal}} < d\left(s_j, b_i\right) \le d_{\max}, \\ 0 & d\left(s_j, b_i\right) > d_{\max} \end{cases} \tag{8}$$

$$\gamma = \frac{p_{\max}}{d_{\max} - d_{\text{ideal}}}, \tag{9}$$

where $p(s_j, b_i)$ denotes the payment of user request which is invoked from b_i and served at s_j. $p_{\max}$ denotes the payment when the requirement of access delay is fulfilled, that is, the access delay $d(s_j, b_i)$ is less than d_{ideal}. If the access delay $d(s_j, b_i)$ exceeds d_{ideal}, the payment decreases as a punishment with a rate of γ. When $d(s_j, b_i)$ becomes larger than $d_{\max}$, the service is not usable at all and, as a result, users will pay nothing to edge providers. Then, the total income can be defined as follows:

$$m_{\text{in}} = \sum_{j=1}^{m} \sum_{i=1}^{n} p(s_j, b_i) \times a_{b_i}(s_j), \tag{10}$$

where m_{in} denotes the total income of edge providers.

E. Problem Statement

In mobile edge computing, edge providers deploy resources at the edge of network and provide services to users. If edge providers want to make more profit, they have to cut cost and increase income. More specifically, if an edge provider wants to make more profit, he has to cut energy consumption to reduce expenditure and provide edge service with lower access delay to increase income. Because energy expenditure has become one of the most significant expenses and sometimes even exceeding hardware cost [39], we mainly consider energy expense as the operation cost. The total profit, denoted as $\mathcal{P}$, is calculated as follows:

$$\mathcal{P} = m_{\text{in}} - m_{\text{cost}}. \tag{11}$$

According to the mentions above, the problem is formulated as follows:

$$Maximize \quad \mathcal{P}, \tag{12}$$

subject to

$$\sum_{j=1}^{m} a_{b_i}(s_j) = 1 \quad (\forall i, 1 \le i \le n), \tag{13}$$

$$\sum_{i=1}^{n} l_{s_j}(b_i) = 1 \quad (\forall j, 1 \le j \le m), \tag{14}$$

$$d(s_j, b_i) \le d_{\max}, \tag{15}$$

$$w_{s_j}(t) \le w_{\max}, \tag{16}$$

where $a_{b_i}(s_j) \in \{0, 1\}$ denotes whether base station b_i is assigned to edge server s_j. $l_{s_j}(b_i) \in \{0, 1\}$ denotes whether edge server s_j is placed at the location of base station b_i. Eq. 15 indicates that the access delay between a base station and its assigned edge node is limited by $d_{\max}$. Eq. 16 indicates that the workload of each edge server should be no more than the upper limit, which is denoted as $w_{\max}$. Above all, the edge server placement problem can be formally stated as follows:

1) find an optimal edge server placement solution $\Omega = (L, A)$,
2) maximizing the total profit in Eq. 11,
3) subject to constraints in Eqs. 13-16.

Theorem 1. *The profit-aware edge server placement problem is NP-Hard.*

Proof. We prove the NP-hardness of profit-aware edge server placement problem by a reduction from set cover problem to it. For a given universe set $\mathcal{Z} = \{z_1, z_2, ..., z_n\}$, a collection of subsets $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_m\}$ of $\mathcal{Z}$ and a size constraint K, the set

cover problem is to find a collection C that satisfying $|C| < K$ and $\bigcup_{i \subseteq C} \mathcal{S}_i = \mathcal{Z}$.

For each z_i in $\mathcal{Z}$, we construct a one-to-one mapping that map z_i to a base station b_i. Similarly, we construct a one-to-one mapping that map a subset $\mathcal{S}_j$ of $\mathcal{Z}$ to a coverage area of edge server. Only one edge server is placed in this coverage area and for all $u_i \in \mathcal{S}_j$, the corresponding b_i will be assigned to the edge server in this area. The collection size constraint K is set to the total number of base stations $|B|$. We can see that a potential solution to the edge server placement problem is also a cover of the universe set $\mathcal{Z}$. Since set cover problem is NP-complete [46], the profit-aware edge server placement problem is NP-hard.

□

IV. Edge Server Placement

In order to solve the problem efficiently, we propose an efficient heuristic algorithm based on PSO. In the following, we will discuss in details.

A. Encoding Scheme

An edge server placement scheme includes edge server placement and base station assignment. Therefore, an encoding scheme should include both of the two relationships. Considering assigning base stations to an edge node is a one-to-many mapping relationship and, as mentioned above all base stations are potential placement locations. A two-dimensional matrix is adopted as the encoding scheme. Leveraging such an encoding scheme brings the following benefits. First, edge server locations are easy to find. Second, the base station assignments are clearly demonstrated. Third, constraint violation judgements are simplified.

TABLE II
Encoding scheme example

	1	2	3	4	5	...	n
1	✓						
2			✓				
3			✓				
4					✓		
5					✓		
...							
n	✓						

For an edge server placement problem with n base stations, the encoding scheme is an $n \times n$ matrix, denoted as M. Each row of the matrix represents a base station and each column represents a potential edge server placement location. Let $M[x][y]$ $(1 \leq x, y \leq n)$ denotes the data located at row x column y in the matrix. If base station b_i is assigned to the edge server placed at base station b_j, $M[i][j]$ will be marked and vice versa. Let's take Table. II as an example. The mark at $M[1][1]$ means base station b_1 is assigned to the edge server placed at base station b_1. The two marks at $M[2][3]$ and $M[3][3]$ means base station b_2 and b_3 are assigned to the edge server placed at base station b_3. Given such an encoding matrix, it is also quite simple to find edge server locations. If $M[x][x]$ is marked, it means base station b_x is chosen to place an edge server.

Because of the constraints Eq. 13 and Eq. 14, placing marks in the matrix should observe the following rules:

- Every row of the matrix must have one mark and at most one mark.
- If column $x(1 \leq x \leq n)$ has one or more marks, column x row x must be marked.

In the evolution phase of the algorithm, the position of a particle is updated in every iteration. The updating operation are very likely to introduce constraint violations. With the help of the encoding scheme, constraint violations can be easily found and eliminated through a deleting and refilling operation. The deleting operation has two steps: 1) Check the encoding row by row. If constraint violations are found, clear the row; 2) If the deleted mark is on the diagonal, clear all the marks in the same column. This is because an edge server is deleted, its assignment relationship with other base stations become invalid. After the two-step delete, only valid marks are remained. Then follows the refilling operation: 1) Check the encoding row by row and find the empty ones; 2) If there is a nearby edge server that can be assigned to, assign this base station to it and mark this relationship in the encoding; 3) If no valid nearby edge server is available, place an edge server at this base station and mark it in the encoding.

B. Edge Server Placement Algorithm

We solve the edge server placement problem based on PSO. PSO has such advantages as faster execution, robustness to control parameters and easy to implement [47]. According to [48], PSO needs less computational effort than genetic algorithm to arrive at the same high quality solutions and, when it comes to computational efficiency, PSO outperforms genetic algorithm with a 99% confidence level. In our edge server placement algorithm, the formulas of PSO are modified as follows:

$$V_i^{t+1} = p_{1i} V_i^t \oplus p_{2i}(X_{i-\text{lbest}}^t \ominus X_i^t) \oplus p_{3i}(X_{\text{gbest}}^t \ominus X_i^t), \quad (17)$$

$$M_i^{t+1} = V_i^{t+1} \otimes (p_{1i} M_i^t \oplus p_{2i} M_{i-\text{lbest}}^t \oplus p_{3i} M_{\text{gbest}}^t), \quad (18)$$

where Eq. 17 denotes the update of particle velocity and Eq. 18 denotes the update of encoding scheme. The most important parameters are M_i^t, X_i^t and V_i^t. M_i^t is the position of particle i at time slot t. It denotes a possible edge server placement solution. The definition is given in Section IV-A. $M_{i-\text{lbest}}^t$ records the local historical value of M_i^t that has the highest profit. M_{gbest}^t records the global historical value of all the M_i^t which has the highest profit.

$X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{in}^t)$ is defined as an n-bit binary vector, where n is the total number of base stations. Each bit represents whether the corresponding base station is chosen to deploy an edge server. X_i^t can be easily got from the diagonal line of M_i^t. $X_{i-\text{lbest}}^t$ and X_{gbest}^t are got from $M_{i-\text{lbest}}^t$ and M_{gbest}^t, respectively.

V_i^t denotes the velocity of particle i at time slot t. It is defined as an n-bit binary vector $V_i^t = (v_{i1}^t, v_{i2}^t, ..., v_{in}^t)$. If the bit $v_{ik}^t = 1(1 \leq k \leq n)$, the $k-th$ column of M_i^t may be changed. V_i^{t+1} is updated from V_i^t according to the following steps. First, calculate $(X_{i\text{lbest}}^t \ominus X_i^t)$ and $(X_{\text{gbest}}^t \ominus X_i^t)$. The $\ominus$ operator is to find the difference between two operands. It is conducted by doing exclusive OR bit by bit. For example, $(1,0,1,0) \ominus (1,1,0,0) = (0,1,1,0)$. Then, V_i^t is updated using $\oplus$ operator. The $\oplus$ operator divides the formula into three parts.

Each part has a probability coefficient p_i, which are defined as follows:

$$p_{1i} = \frac{\mathcal{P}_i}{\mathcal{P}_i + \mathcal{P}_{i-\text{lbest}} + \mathcal{P}_{\text{gbest}}}, \tag{19}$$

$$p_{2i} = \frac{\mathcal{P}_{i-\text{lbest}}}{\mathcal{P}_i + \mathcal{P}_{i-\text{lbest}} + \mathcal{P}_{\text{gbest}}}, \tag{20}$$

$$p_{3i} = \frac{\mathcal{P}_{\text{gbest}}}{\mathcal{P}_i + \mathcal{P}_{i-\text{lbest}} + \mathcal{P}_{\text{gbest}}}, \tag{21}$$

where $\mathcal{P}_i$ is the total profit of M_i^t, $\mathcal{P}_{i-\text{lbest}}$ is the total profit of $M_{i-\text{lbest}}^t$ and $\mathcal{P}_{\text{gbest}}$ is the total profit of M_{gbest}^t. It is obvious that the more profit, the greater the value of its probability coefficient. The $\oplus$ operator updates V_i^t bit by bit using roulette strategy. Take $(1,\eta,0) = p_{1i}(1,a,0) \oplus p_{2i}(1,b,0) \oplus p_{3i}(1,c,0)$ as an example. η in the equation is an uncertain bit:

$$\eta = \begin{cases} a, & 0 \le rand \le p_{1i} \\ b, & p_{1i} < rand \le p_{1i} + p_{2i} \\ c, & p_{1i} + p_{2i} < rand \le 1 \end{cases}, \tag{22}$$

where the value of a, b, c is 0 or 1 and $rand$ is a random number between 0 and 1. Note that $p_{i1} + p_{i2} + p_{i3} = 1$.

With an updated V_i^{t+1}, M_i^t is updated next. Operator $\otimes$ defines the particle position updating operation. The operation has three steps. 1) Decide which column to update. For $V_i^{t+1} = (v_{i1}^{t+1}, v_{i2}^{t+1}, ..., v_{in}^{t+1})$, if $v_{ik}^{t+1} = 1$, the k-th column of M_i^t will be updated. 2) Update the encoding matrix. The chosen columns of M_i^t will be replaced by the corresponding column calculated from $(p_{1i}M_i^t \oplus p_{2i}M_{i-\text{lbest}} \oplus p_{3i}M_{\text{gbest}})$. The $\oplus$ here is the same as we mentioned before. 3) Adjust the coding matrix so that it does not violate the constraints.

Note that, edge server placement includes selecting edge server locations and base station assignment. In the process of base station assignment, we introduce a weight value q as a guidance.

$$q = \frac{d_{\max}}{d(s_j, b_i)} + \frac{w_{b_i}}{\sum_{b_k \in C(s_j)} w_{b_k}}, \tag{23}$$

where $d_{\max}$ is the access delay constraint. $d(s_j, b_i)$ denotes the access delay between s_j and b_i. $w_{b_i} = \int_{t_1}^{t_2} w_{b_i}(t)\,dt$ denotes the accumulative workload of b_i. $\sum_{b_k \in C(s_j)} w_{b_k}$ calculates the total accumulative workload of all base stations within the coverage of s_j, where $C(s_j)$ denotes the set of base stations within the coverage of s_j. Base stations that are closer to edge nodes or have a larger workload will have a higher q value. That is, it will have a higher priority to be assigned. This will prevent following awkward situations: 1) When assign base stations with heavy workload, the nearest edge server does not have enough resources to support the assignment even if there is still quite a few resources left in the edge server. Because its computation resources have already been occupied to serve large number of base stations with light workload; 2) Faraway base stations within the coverage are assigned to the edge server while base stations at core of the serving area have no choice but forwarding service requests to other edge nodes. Both situations fail to allocate resources properly, which will inevitably lead to poor access delay and energy waste.

By introducing q value based initialization and the two-dimension encoding scheme, the edge server placement algorithm runs as Algorithm 1. In order to distinguish it from the original PSO, we use QPSO to refer to our modified algorithm in the following paragraphs.

Algorithm 1: Edge Server Placement

Input: data set of base stations; $d_{\max}$ of edge server; population size of particle swarm N; algorithm iteration number $\mathcal{T}$;
Output: edge server placement scheme

```
Initialize the N particles;
Initialize M_{i-lbest}^t of all particles;
Initialize M_{gbest}^t;
while t < T do
    foreach particle of total N particles do
        update velocity of each particle according to Eq. 17;
        update position of each particle according to Eq. 18;
        find constraint violations;
        eliminate constraint violations via deleting and refilling;
    update M_{i-lbest}^{t+1};
    update M_{gbest}^{t+1};
    t ← t +1;
return M_{gbest}^{T}.
```

V. Performance Evaluation

In this section, we conduct various experiments to evaluate the performance of the profit-aware edge server placement algorithm and compare it with other algorithms.

A. Experiment Setup

The experiment is conducted by simulating edge server placement in the whole city of Shanghai. We use Shanghai Telecom base station dataset[2] to build a simulation environment close to reality. The dataset consists of the locations of total 3233 base stations and the Internet access log of users. If a user gets access to the Internet, we can learn when it starts, when it ends and which base station the user connects to. Table. III is an example of the processed data. All base stations in the table are selected randomly.

We use this dataset as an input to provide temporal and spatial distribution of user workload. The dataset records the operation status of base stations in Shanghai for 15 consecutive days. So we mainly study the total energy consumption and working state of all the base stations in a period of 15 days. In addition, considering the fact that it is almost impossible to get the real-time CPU utilization efficiency at each moment, we choose working minutes as the metric of workload. Therefore, the CPU utilization efficiency can be calculated indirectly, and we can get the total energy consumption of one edge server in 15 days. Then, we generate the network topology based on base station locations provided in the dataset. At first, we link adjacent base stations to form one-hop relationships. Then we use Floyd algorithm [49] to generate the whole topology.

However, having the topology is not enough. We cannot get access delay for each user request. To tackle this problem, we build an emulator called mini5Gedge to acquire access delay of a link between user equipments and edge servers

[2]http://sguangwang.com/TelecomDataset.html

TABLE III
INFORMATION OF A PART OF BASE STATIONS

ID	User number	Latitude	Longitude	Workload(min)
7	5	31.2377	121.3825	5556
16	165	31.4837	121.2748	215632
193	367	31.0154	121.1548	454070
446	212	31.2815	121.5582	252115
593	164	31.3068	121.5196	209063
677	286	31.3011	121.1778	378484
753	202	31.4534	121.2564	253360
833	10	31.2567	121.4387	13990
1005	24	31.3971	121.5077	32982
1141	17	31.1860	121.4487	17978
1325	53	31.2479	121.5134	67468

through UPF. As is shown in Fig. 3, we implement user plane network functions to enable a real connection between user equipments and edge servers. In this emulator, the wireless communications between user equipments and radio access network (RAN) are replaced by Ethernet connections. Since we mainly focus on the backhaul links instead of the wireless connection, this simplification can help us prevent the complication brought by the wireless communication protocol stack. When a user equipment wants to get access to the edge server, the emulator will build up a GTP-U tunnel for it [4]. We use I-UPF and A-UPF to refer to UPFs playing different role in a tunnel. The I-UPF act as an intermediate node in the process of building up a tunnel. It serves as an uplink classifier and forward traffics to a proper A-UPF. The A-UPF serves as a packet data unit session anchor. The tunnel starts from RAN, passing through I-UPF and end at A-UPF. We leverage this emulator to get the access delay by measuring round-trip time under different circumstances.

We designed two experiments: 1) keep the number of base stations fixed at 1100 while change the $d_{\max}$ of edge server from 2 *ms* to 8 *ms*; 2) keep the $d_{\max}$ of edge server equals 5 *ms* while change the total number of base stations from 300 to 1300. In order to simplify the experiment, we set $d_{\text{ideal}} = 0.5d_{\max}$. The initial population size in our PSO based algorithms is set to 40 [50], and the maximum iteration number is set to 400 according to our another experiment in Section V-E.

We choose Dell PowerEdge R730 as the edge server. It has an Intel Xeon E5-2620 v4 CPU. The main frequency of CPU is 2.1 GHz and the RAM is 32 GB. Its full state power is 495W. According to the survey result in [39], the idle power of edge server account for 60% of the full state power. The electricity price[3] is 0.925 RMB per kWh. The maximum payment $p_{\max}$ is 0.0016 RMB per minute according to the price of Tencent Cloud S1[4].

B. Benchmark Algorithms

We compare our QPSO with other placement algorithms in terms of total profit, average access delay, energy consumption and hardware resource utilization efficiency. The algorithms are as follows:

1) **TopFirst.** In every loop, base station with the heaviest workload is selected to place edge server. Base stations in the coverage area satisfying constraints are assigned to it. This process is repeated until all the base stations are assigned.
2) **Random.** This algorithm is to select base stations randomly to place edge servers and assign base stations satisfying constraints to them.
3) **Greedy.** In the first iteration, base stations merge with its nearest neighbors to form a group. Then the groups merge with its nearest group to form larger groups in the following iterations. The total workload of a group is limited by $w_{\max}$, and the base station with the lowest delay between other base stations in the group will be chosen to place an edge server.
4) **PSO.** The origin PSO [32] without using q value.

[3]http://fgw.sh.gov.cn/jgl/20190329/0025-35583.html
[4]https://buy.cloud.tencent.com/price/cvm/overview

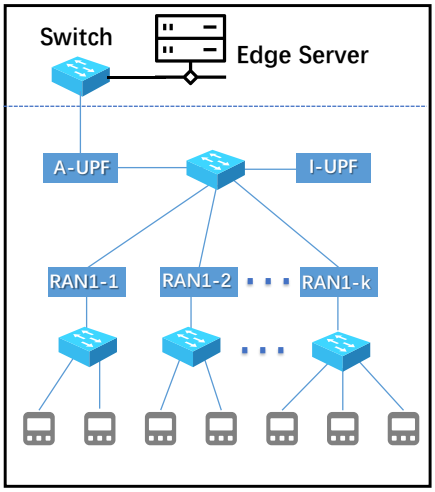

Fig. 3. System design of mini5Gedge.

C. Performance with Varying Maximum Delay

Figs. 4-7 show the performance of algorithms when $d_{\max}$ changes from 2 *ms* to 8 *ms*. The total number of base stations is 1100. For PSO and QPSO, the population size $N = 40$. The maximum iteration number $\mathcal{T} = 400$.

1) Total Profit: As is shown in Fig. 4, QPSO achieves the highest profit in most cases. When $d_{\max} = 2$ *ms*, only PSO and QPSO has a positive profit value. Greedy, Random and TopFirst are even in deficit. The low profit mainly result from high energy consumption(shown in Fig. 6). $d_{\max} = 2$ *ms* is such a strict constraint that all the algorithms have no choice but deploy far more edge servers. The large amount of energy consumption lead to heavy operation cost. As $d_{\max}$ increases, all algorithms witness a rapidly growing profit. Since $d_{\max} = 4$ *ms*, QPSO earns the highest profit. Then follows Greedy and PSO. TopFirst has the lowest profit. When $d_{\max}$ is larger than 5 *ms*, the growth nearly stops. In this stage, QPSO earns a profit on average 17.79% larger than Greedy and 27.09% larger than PSO. TopFirst has the lowest profit. Its profit is on average only 54.70% of QPSO's.

2) Access Delay: Figs. 5 shows average access delay of all algorithms as $d_{\max}$ increases. In general, QPSO achieves the lowest access delay in most cases. Besides, as $d_{\max}$ gets larger, the access delay of algorithms all increases and the gap between algorithms becomes larger. When $d_{\max} = 2$ *ms*, the average access delay of QPSO is 16.12%, 10.02%, 5.47%

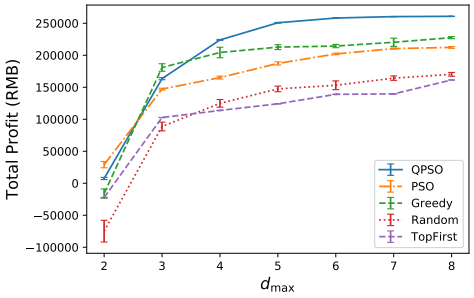

Fig. 4. Total profit for the edge server placement with respect to the $d_{\max}$ of edge server.

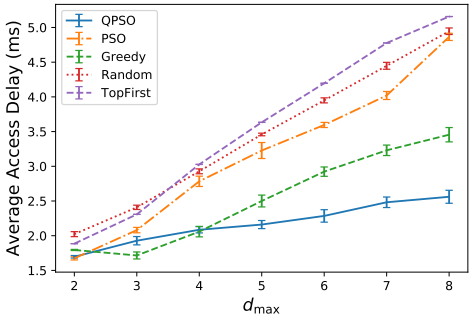

Fig. 5. Average access delay for the edge server placement with respect to the $d_{\max}$ of edge server.

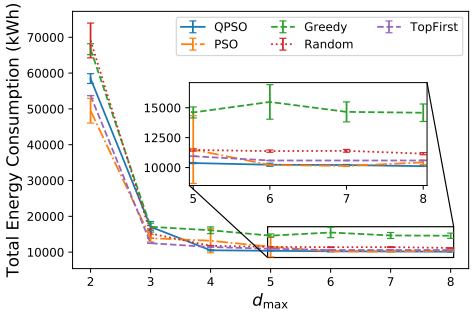

Fig. 6. Energy consumption for the edge server placement with respect to the $d_{\max}$ of edge server.

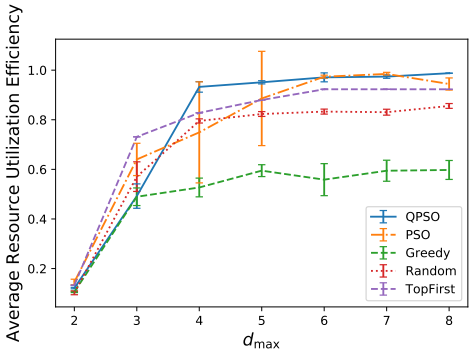

Fig. 7. Average resource utilization efficiency for the edge server placement with respect to the $d_{\max}$ of edge server.

less than that of Random, TopFirst and Greedy, respectively. Compared with PSO, the access delay of QPSO is only 1.36% more than that of PSO. However, when $d_{\max} = 8$ *ms*, QPSO has far less access delay than other algorithms. Its average access delay is 25.88% less than that of Greedy(having the second lowest delay) and 50.35% less than TopFirst(having the largest delay). In the process of $d_{\max}$ increasing from 2 *ms* to 8 *ms*, the coverage area of one edge server gets larger. Greedy follows a low delay first strategy and achieves the second lowest access delay. TopFirst place edge servers at base stations with the heaviest workload, which does not necessarily have low access delay with UPF. Neither QPSO nor PSO optimizes access delay directly. However, with the guidance of q value, QPSO assigns base stations to edge servers in a more proper way. This prevents base stations being assigned to a further edge server because of insufficient computing resources left at the nearest edge server.

3) Total Energy Consumption: As is shown in Fig. 6, with $d_{\max} = 2$ *ms* all algorithms have tremendous amount of average energy consumption, ranging from 49497.78 kWh (PSO) to 69111.66 kWh (Random). Then, a rapid drop happens. The energy consumption of five algorithms reduces at least 70.79% (QPSO) from $d_{\max} = 2$ *ms* to $d_{\max} = 3$ *ms*. As $d_{\max}$ becomes larger than 4 *ms*, the reduction of total energy consumption slow down. Because Greedy follows a delay first strategy, it has the highest energy consumption. QPSO and PSO are the top two energy-saving algorithms. In the range from $d_{\max} = 4$ *ms* to $d_{\max} = 8$ *ms*, QPSO consumes on average 31.70% less energy than Greedy. A similar tendency occurs in Fig. 7, where average resource utilization efficiency goes up sharply from $d_{\max} = 2$ *ms* to $d_{\max} = 4$ *ms* and slows down afterwards. This is because when $d_{\max}$ is less than 4 *ms*, the number of base stations assigned to one edge server is small because the size of coverage area is limited by $d_{\max}$. Each edge server can only serve a few base stations in a small area, even if the total workload of this area is far away from the maximum workload of the edge server. Large number of edge servers are placed to prevent coverage holes even though they work in low efficiency. However, when $d_{\max}$ gets larger than 4 *ms*, the resource utilization efficiency approaches the upper limit. The reduction of edge server number slows down greatly.

4) Summary: In summary, QPSO stands out in terms of making the highest profit. It also achieves the lowest access delay. Although Greedy has the second shortest access delay, it is at the cost of a huge extra energy consumption. PSO succeeds in reducing energy consumption but fails to guarantee low access delay. By comparing QPSO with benchmark algorithms, it is obvious that QPSO has a better trade off between energy consumption and access delay. As a result, only QPSO keeps access delay less than d_{ideal} when $d_{\max}$ is larger than 5 *ms*. This means only QPSO can acquire full payment according to SLA.

D. Performance with Varying Number of Base Stations

Figs. 8-11 show the performance of algorithms when the total number of base stations changes from 300 to 1300. The objective is to evaluate the performance of algorithms under different total workload of the city. In this experiment, the $d_{\max}$ is 5 *ms*. For PSO and QPSO, the initial population size $N = 40$. The maximum iteration number $\mathcal{T} = 400$.

1) Total Profit: Fig. 8 shows the performances in terms of profit. QPSO, Greedy and PSO earn far more profit than Random and TopFirst and the gap becomes larger as the number of base stations increases. The profit of QPSO is

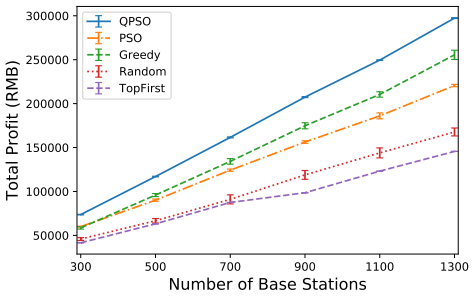

Fig. 8. Total profit for the edge server placement with respect to the number of base stations.

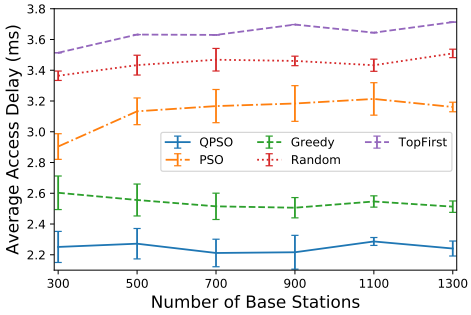

Fig. 9. Average access delay for the edge server placement with respect to the number of base stations.

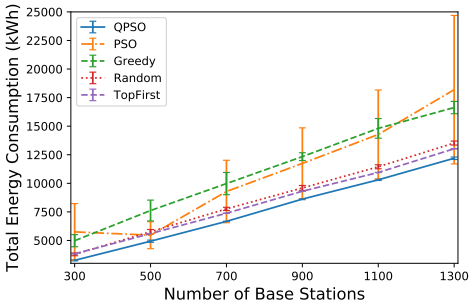

Fig. 10. Energy consumption for the edge server placement with respect to the number of base stations.

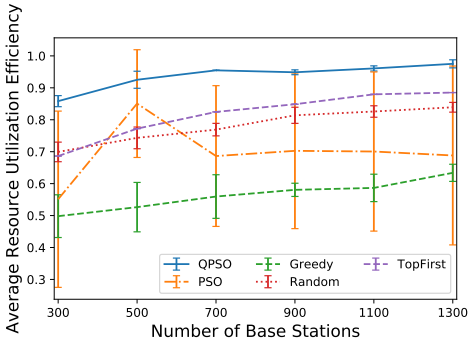

Fig. 11. Average resource utilization efficiency for the edge server placement with respect to the number of base stations.

on average 73.16% more than that of Random and 94.08% more than that TopFirst. For the top three algorithms, QPSO performs the best, then follows Greedy. The profit of QPSO is on average 20.28% more than that of Greedy and 30.74% more than that of PSO. The total profit of the five algorithms rises linearly. This is because the utilization efficiency per edge server changes very little (as shown in Fig. 11), the extra workload brought by the growth of base station numbers mainly result in the increase of total edge server number.

2) Access Delay: As for access delay(shown in Fig. 9), when total number of base station increases, the performance of Random almost keeps unchanged while PSO and TopFirst witness a slight increase. The access delay of Greedy goes down slightly. QPSO keeps access delay under a pretty low level, on average 11.55% less than Greedy and 28.08% less than PSO. The reasons for such a delay change are as follows. TopFirst place edge servers at base stations with the heaviest workload. However, these base stations may have a large access delay with the nearest UPF. Therefore, TopFirst performs even worse than Random. PSO optimizes the access delay indirectly during the optimization of total profit. Greedy chooses base station with the lowest access delay in every iteration, but neglect the workload difference between base stations. Only QPSO leverages a q value to jointly consider access delay and total workload constraints and achieve the lowest access delay.

3) Total Energy Consumption: Figs. 10 shows how total energy consumption changes. As number of base stations increases, the total energy consumption of five algorithms all go up. That is because the increasing base stations bring more workload. In this process, PSO has the highest average total energy consumption. Then follows Greedy, with only a slight gap. QPSO keeps the lowest energy consumption, on average 28.08% less than PSO and 31.69% less than Greedy. The reason why QPSO saves more energy lies in resource utilization efficiency. As is shown in Fig. 11, with different number of base stations, QPSO keeps the highest resource utilization efficiency. As a result, it can prevent edge servers working in idle state and reduce total number of edge server, which contributes to energy saving. It is worth noting that the standard deviation of PSO is far greater than other algorithms. This is because the origin PSO only optimize total profit. Although its total profit converges well, the algorithm achieves the optimization by pursuing either extreme low access delay or extreme low energy consumption which result in the significant variation on energy consumption. On the contrary, QPSO achieves a better trade-off and, as a result, has the highest total profit.

4) Summary: QPSO performs best in terms of increasing profit. It keeps the best trade-off between access delay reduction and energy saving. By jointly consider access delay and workload distribution, it achieves the lowest access delay. Note that, QPSO is the only one algorithm that keep the access delay within d_{ideal}. Thus, it has full payment in SLA. In addition, it can find an edge server placement scheme with the lowest energy consumption by means of optimizing base station assignment scheme to improve utilization ratio of edge servers.

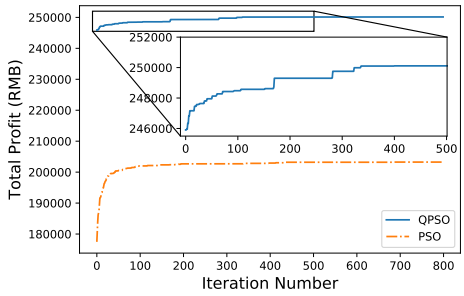

Fig. 12. Total profit for edge server placement with respect to iteration number.

E. Study on Iteration Number

In order to find out the influence of the maximum iteration number on QPSO and PSO, we conducted another experiment. In this experiment, we record the total profit of every iteration with following parameters: $d_{\max} = 5\ ms$ and total base station number = 1100.

The result of the experiment is shown in Fig. 12. To make it clearer how the total profit of QPSO changes with different iteration number, we enlarge the line of QPSO when iteration number is less than 500. The total profit of both QPSO and PSO goes up quickly at first and the increase slows down gradually after the iteration number reaches 200. Finally, it remains almost unchanged when the iteration number is larger than 400. That is, the algorithms have converged with 400 iterations. Based on the result, the maximum iteration number is set to 400 in Section V-C and Section V-D.

VI. Conclusion and Future Work

In this paper, we study the problem of edge server placement from the perspective of edge providers. First, we propose an edge server placement model, which takes UPF into consideration in the access delay model for the first time. Then, we introduce SLA model to help balance the trade-off between access delay and energy consumption. We put forward QPSO, which introduces parameter q, to help assign base stations to edge nodes in a more reasonable way. The performance of our algorithm is evaluated by a series of experiments based on Shanghai Telecom base station dataset. The experiment results show that QPSO can increase more than 17.79% profit while achieve a good trade-off between access delay and energy consumption.

This work is only the first step of edge deployment. In future work, we will study budget optimization which includes construction cost, server purchasing price, energy cost and so on. In addition, we will study the dynamic management of edge resources. As the coverage area of one edge node is not big, the service requirement in this area is relatively dynamic. First, the service requirements of users change over time. Second, user movement makes it hard to provide service by only one edge node. Therefore, the dynamic deployment and the migration of services are new challenges in MEC. We will focus on these problems in future work.

Acknowledgment

This work was supported in part by National Key R&D Program of China (2020YFB1805502), NSFC (61922017 and 61921003), the Open Research Fund of Key Laboratory of Space Utilization, Chinese Academy of Sciences (No.LSU-KFJJ-2019-03).

References

[1] X. Deng, Z. Tang, L. T. Yang, M. Lin, and B. Wang, "Confident Information Coverage Hole Healing in Hybrid Industrial Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2220–2229, 2018.

[2] J. Xu, K. Ota, and M. Dong, "Big data on the fly: Uav-mounted mobile edge computing for disaster management," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2620–2630, 2020.

[3] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Energy-efficient resource allocation in software-defined mobile networks with mobile edge computing and caching," in *Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2017)*, May 2017, pp. 121–126.

[4] 3GPP, "General packet radio system (GPRS) tunnelling protocol user plane (GTPv1-U)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.281. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/29_series/29.281/

[5] 3GPP, "GPRS tunnelling protocol (GTP) across the Gn and Gp interface," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.060. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/29_series/29.060/

[6] 3GPP, "System architecture for the 5G System," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/

[7] 3GPP, "5G system; interworking between 5g network and external data networks," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.561. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/29_series/29.561/

[8] Y. Li and S. Wang, "An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing," in *Proceedings of IEEE International Conference on Edge Computing (EDGE 2018)*, Jul 2018, pp. 66–73.

[9] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in Wireless Metropolitan Area Networks," in *Proceedings of the 40th IEEE Conference on Local Computer Networks (LCN 2015)*, 2015, pp. 570–578.

[10] 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, Jun 2018.

[11] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal Placement of Cloudlets for Access Delay Minimization in SDN-Based Internet of Things Networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334–1344, Apr 2018.

[12] Q. Fan and N. Ansari, "Cost Aware cloudlet Placement for big data processing at the edge," in *Proceedings of IEEE International Conference on Communications (ICC 2017)*, May 2017, pp. 1–6.

[13] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge Provisioning with Flexible Server Placement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1031–1045, Apr 2017.

[14] H. D. Chantre and N. L. S. da Fonseca, "Multi-Objective Optimization for Edge Device Placement and Reliable Broadcasting in 5G NFV-Based Small Cell Networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2304–2317, Oct 2018.

[15] L. Chen, S. Zhou, and J. Xu, "Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, Aug 2018.

[16] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy Aware Offloading for Competing Users on a Shared Communication Channel," *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 87–96, Jan 2017.

[17] Y. Geng, Y. Yang, and G. Cao, "Energy-Efficient Computation Offloading for Multicore-Based Mobile Devices," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM 2018)*, Apr 2018, pp. 46–54.

[18] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, Feb 2019.
[19] S. Kosta, A. Aucinas, Pan Hui, R. Mortier, and Xinwen Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM 2012)*, Mar 2012, pp. 945–953.
[20] Y. Wang and Y. Xia, "Energy optimal VM placement in the cloud," in *Proceedings of the 9th IEEE International Conference on Cloud Computing (CLOUD 2016)*, 2016, pp. 84–91.
[21] A. Kaur and M. Kalra, "Energy optimized vm placement in cloud environment," in *Proceedings of the 6th International Conference - Cloud System and Big Data Engineering (Confluence 2016)*, 2016, pp. 141–145.
[22] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," in *Proceedings of the 1st International Conference on Cloud Computing (CloudCom 2009)*, 2009, pp. 24–44.
[23] H. Yao, H. Li, C. Liu, M. Xiong, D. Zeng, and G. Li, "Joint Optimization of VM Placement and Rule Placement towards Energy Efficient Software-Defined Data Centers," in *Proceedings of the IEEE International Conference on Computer and Information Technology (CIT 2017)*, 2017, pp. 204–209.
[24] A. Santoyo-Gonzalez and C. Cervello-Pastor, "Edge Nodes Infrastructure Placement Parameters for 5G Networks," in *In Proceedings of IEEE Conference on Standards for Communications and Networking (CSCN 2018)*. Paris, France: IEEE, Oct 2018, pp. 1–6.
[25] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 926–937, Jul 2019.
[26] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-Effective Edge Server Placement in Wireless Metropolitan Area Networks," *Sensors*, vol. 19, no. 1, p. 32, Jan 2019.
[27] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet Placement and Task Allocation in Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5853–5863, Jun 2019.
[28] G. Manasvi, A. Chakraborty, and B. S. Manoj, "Social Network Aware Dynamic Edge Server Placement for Next-Generation Cellular Networks," in *Proceedings of International Conference on COMmunication Systems & NETworkS (COMSNETS 2020)*. Bengaluru, India: IEEE, Jan 2020, pp. 499–502.
[29] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, "Robustness-oriented k Edge Server Placement," in *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID 2020)*. Melbourne, Australia: IEEE, May 2020, pp. 81–90.
[30] W. Huang, K. Ota, M. Dong, T. Wang, S. Zhang, and J. Zhang, "Result return aware offloading scheme in vehicular edge networks for IoT," *Computer Communications*, vol. 164, pp. 201–214, 2020.
[31] S. Kaur and S. Bawa, "A review on energy aware vm placement and consolidation techniques," in *Proceedings of the International Conference on Inventive Computation Technologies (ICICT 2016)*, 2016, pp. 1–7.
[32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN 1995)*, 1995, pp. 1942–1948.
[33] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3658–3668, Nov 2017.
[34] L. N. T. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient Computation Offloading in Multi-Tier Multi-Access Edge Computing Systems: A Particle Swarm Optimization Approach," *Applied Sciences*, vol. 10, no. 1, pp. 203–219, Jan 2020.
[35] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-Optimized Partial Computation Offloading in Mobile-Edge Computing With Genetic Simulated-Annealing-Based Particle Swarm Optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, Mar 2021.
[36] K. E. Purushothaman and V. Nagarajan, "Multiobjective optimization based on self-organizing Particle Swarm Optimization algorithm for massive MIMO 5G wireless network," *International Journal of Communication Systems*, vol. 34, no. 4, p. e4725, 2021.
[37] Z. Lv, L. Wang, Z. Han, J. Zhao, and W. Wang, "Surrogate-assisted particle swarm optimization algorithm with Pareto active learning for expensive multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 838–849, May 2019.
[38] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang, and W. A. Chaovalitwongse, "Comprehensive Learning Particle Swarm Optimization Algorithm With Local Search for Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 718–731, Aug 2019.
[39] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, Firstquarter 2016.
[40] X. Fan, W. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th International Symposium on Computer Architecture (ISCA 2007)*, 2007, pp. 13–23.
[41] V. Gupta, R. Nathuji, and K. Schwan, "An analysis of power reduction in datacenters using heterogeneous chip multiprocessors," *Acm Sigmetrics Performance Evaluation Review*, vol. 39, no. 3, pp. 87–91, 2013.
[42] G. Dasgupta, A. Sharma, A. Verma, A. Neogi, and R. Kothari, "Workload Management for Power Efficiency in Virtualized Data Centers," *Communications of the Acm*, vol. 54, no. 7, pp. 131–141, 2011.
[43] G. Chen, S. Nath, S. Nath, S. Nath, F. Zhao, F. Zhao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proceedings of the 5th Usenix Symposium on Networked Systems Design and Implementation (NSDI 2008)*, 2008, pp. 337–350.
[44] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *Proceedings of the 19th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2013)*, 2013, pp. 102–109.
[45] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes, "Service contracts and aggregate utility functions," in *Proceedings of IEEE International Conference on High Performance Distributed Computing (HPDC 2006)*, Jun 2006, pp. 119–131.
[46] R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds. Boston, MA: Springer US, 1972, pp. 85–103.
[47] K. Y. Lee and J. b. Park, "Application of particle swarm optimization to economic dispatch problem: Advantages and disadvantages," in *Proceedings of the IEEE PES Power Systems Conference and Exposition (PSCE 2006)*, 2006, pp. 188–192.
[48] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamicsand Materials Conference (SDM 2005)*, 2005.
[49] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, Jun 1962.
[50] C. Lei, "The study on dynamic population size improvements for classical particle swarm optimization," in *Proceedings of the International Conference on Computer Science and Network Technology (ICCSNT 2011)*, 2011, pp. 430–433.

Yuanzhe Li received a Bachelor degree of Engineering degree in communication engineering from Beijing University of Posts and Telecommunications(BUPT), in 2016. Currently, he is a Ph.D. candidate at the State Key Laboratory of Networking and Switching Technology, BUPT. His research interests include mobile edge computing, cloud computing and service computing.

Ao Zhou received the PH.D. degrees in Beijing University of Posts and Telecommunications, Beijing, China, in 2015. She is currently an Associate Professor with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has published 20+ research papers. She played a key role at many international conferences. Her research interests include Cloud Computing and Edge Computing.

Xiao Ma received her Ph.D. degree in Department of Computer Science and Technology from Tsinghua University, Beijing, China, in 2018. She is currently a postdoctoral fellow at the State Key Laboratory of Networking and Switching Technology, BUPT. From October 2016 to April 2017, she visited the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include mobile cloud computing and mobile edge computing.

Shangguang Wang (Senior Member, IEEE) received his Ph.D. degree in computer science and engineering at Beijing University of Posts and Telecommunications in 2011. He is currently a Professor at the School of Computing, Beijing University of Posts and Telecommunications, China. He is a Vice-Director of the State Key Laboratory of Networking and Switching Technology. He has published more than 150 papers, and his research interests include service computing, cloud computing, and mobile edge computing.

He served as General Chairs or TPC Chairs of IEEE EDGE 2020, IEEE CLOUD 2020, IEEE SAGC 2020, IEEE EDGE 2018, and IEEE ICFCE 2017, etc., and Vice-Chair of IEEE Technical Committee on Services Computing (2015-2018). He is currently serving as Executive Vice-Chair of IEEE Technical Committee on Services Computing (2021-), and Vice-Chair of IEEE Technical Committee on Cloud Computing (2020-). He is a senior member of the IEEE.