Preference-Aware Edge Server Placement in the Internet of Things

Yuanyi Chen, Yihao Lin, Zengwei Zheng, Peng Yu, Jiaxing Shen, and Minyi Guo, *Fellow, IEEE*

***Abstract*—While it is well understood that edge computing can significantly facilitate IoT-related applications by deploying edge servers close to IoT devices, it also faces many challenges with numerous IoT devices connected and interacted. One of the most important issues is how to efficiently deploy edge servers under a certain budget with the explosive growth of data scale and user base. Existing studies for edge server placement fail to consider user's query preferences since individual users may be interested in events in particular regions and are keen to receive up-to-date data streams that originate in regions of interest. In this article, we present a preference-aware edge server placement approach that offers better workload distribution in terms of both minimizing query latency and balancing the load of edge servers. To achieve this, we formulate edge server placement with multiobjective optimization as a *p*-center problem and design two progressive approaches. We first propose quadratic integer programming (QIP) for small-scale data sets. Since the *p*-center problem is an NP-hard problem, we thus propose a heuristic algorithm named TAKG (TAbu search with *K*-means and Genetic algorithm) for large-scale data sets. To evaluate the utility of the proposed models, we have conducted a comprehensive evaluation on a large data set that is collected by more than 1900 IoT devices during 30 days. Experimental results indicate our approaches outperform all baselines significantly in terms of both query latency and load balancing.**

***Index Terms*—Edge server deployment, Internet of Things (IoT), multiobjective optimization, query latency, user preference.**

I. Introduction

RECENT years have witnessed the rapid advancement of Internet of Things (IoT) by connecting various physical things embedded with sensing, communication and computing capabilities (e.g., mobile phones, wallets, and key chains). IoT has become the enabler of numerous intelligent applications, including intelligent transportation [1], [2], IoT service recommendation [3], [4], acoustic gesture recognition [5], blockchain applications in mobile IoT [6], [7], and WiFi-based group detection [8], [9]. In most applications, end users need to collect certain data types at particular locations and during specific times which leads to the accumulation of massive IoT data [10]–[12].

Unfortunately, most IoT devices especially those with limited resources are not capable of handling such large amounts of data in real time [13]. Additionally, the massive data transferred from the IoT data source to the remote cloud center will bring many challenges, such as: 1) increasing the capacity pressure of the backhaul link and even lead to network blocking, data loss and other errors; 2) putting a lot of storage pressure on cloud server because there is a large amount of meaningless, erroneous or duplicate data taking up efficient and limited storage space, thus reducing productivity; and 3) augmenting the computing load of the cloud computing platform when a large amount of data get in simultaneously. These challenges motivate a move toward an edge computing approach that facilitates the collection and cleaning of data closer to IoT devices, which increase the processing efficiency and reduce cloud server storage costs by transferring only relevant information [11], [14], [15]. As more and more devices access the IoT, efficient edge server deployment scheme need to be developed to process the spatiotemporality of IoT data and resources to support high scalability [16], [17].

Existing studies (for a review see Section II) for edge server placement are capable of efficiently deploying edge server under a certain budget with different optimization goals, such as minimizing latency [18], [19] or deploy cost [20], [21], reducing energy consumption [22] and the robustness of edge server network [23], [24]. However, the workload of edge servers varies dramatically over time due to individual users may be interested in data in particular regions and are keen to receive up-to-date data streams from their interested regions. Fig. 1 depicts the distribution of query frequency of users to a specific station using a real-world data set of California's freeway, more details of the data sets are reported in Section V-A. From this figure, we observe that the number of base stations with a particularly high number of queries is small while most base stations receive few retrieval requests. This phenomenon inspires us to pay more attention to the edge servers that users are highly interested in to prevent them from overloading or even crashing due to the large number of work requests during certain periods of time.

In this article, we present quadratic integer programming (QIP), a preference-aware edge server placement method for IoT-related applications. The design objective of QIP

Manuscript received February 20, 2021; accepted May 4, 2021. Date of publication May 11, 2021; date of current version January 7, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61802343 and Grant 62072402; in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LGF19F020019 and Grant LGN20F020003; in part by the Hangzhou Science and Technology Bureau under Grant 20191203B37; and in part by the Intelligent Plant Factory of Zhejiang Province Engineering Lab. *(Yuanyi Chen and Yihao Lin are co-first authors.) (Corresponding author: Zengwei Zheng.)*

Yuanyi Chen, Yihao Lin, Zengwei Zheng, and Peng Yu are with the Department of Computer Science and Computing, Zhejiang University City College, Hangzhou 310015, China (e-mail: zhengzw@zucc.edu.cn).

Jiaxing Shen is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

Minyi Guo is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.

Digital Object Identifier 10.1109/JIOT.2021.3079328

Fig. 1. Frequency of query time.

is to accomplish better workload distribution in terms of minimizing latency and balancing the query load of edge servers. To accomplish these goals, we propose a handful of new techniques. First, we refine the original problem into a multiobjective optimization problem with NP-hard complexity after a lot of work of mathematical modeling. Second, we propose a QIP solution to find the optimal edge server placement scheme for small-scale data set. For large-scale data set, we propose a heuristic algorithm named TAKG (TAbu search with K-means and Genetic algorithm) to find an approximate edge server placement scheme based on multiple iterations. The main contributions of this article are three folds.

1) According to the working process of the edge server, the optimization goal is expanded into three aspects, memory balancing, query balancing, and average access delay. We formulate the problem as a multiobjective optimization problem, and refer to the method of p-center problem.
2) We adopt QIP to achieve accurate solution on small-scale data set and design an algorithm inspired by TABU search on large-scale data set for approximate solution.
3) We evaluate our approaches based on a real world data set collected by more than 1900 users over 30 days. The results show the advantages of our approaches compare with state-of-art baselines.

The remainder of this article is organized as follows. In Section II, we show related work about edge server placement. In Section III, we present the problem description and prove the NP-hardness. In Section IV, we propose our methods and analyze the complexity. In Section V, we introduce the data set and discuss the experimental results comparing to other methods. Finally, we present our conclusion and future work in Section VI.

II. Related Work

Unlike the cloud computing model, edge computing emphasizes that data and requests generated by user devices are analyzed and processed at the edge of the network [25]. As the number of edge servers increases, how to place the edge servers is a key issue. Earlier studies [26], [27] usually assumed that the edge servers they need have been properly deployed and merely considered how to efficiently use the caching capabilities of the servers to store and retrieve data quickly and accurately, another literature [28], [29] focused on cloudlet placement problem using many wireless APs. For example, the study [28] deployed cloudlets on the network and allocate each requested task to cloudlets with the minimum total energy consumption without violating each task's delay requirement, and further proved this problem is NP-hard and propose a Benders decomposition-based solution. Similarly, Bhatta and Mashayekhy [29] proposed a cost-aware approach to deploy a set of heterogeneous cloudlets in a region under the condition of meeting user's latency requirements.

Recently, a few efforts have been made to efficiently deploy edge server under a certain budget and different optimization goals. Actually, it is impossible to deploy enough edge servers everywhere since the budgets of edge computing service provider are always limited. To address this challenge, the studies [20], [21] investigated how to deploy edge servers effectively and economically by minimizing the number of edge servers while ensuring some QoS requirements. Specifically, Zeng *et al.* [20] transformed the cost-effective edge server placement into the minimum dominating set problem in graph theory, while the work [21] formulated the problem as an integer linear programming problem and solved with a greedy algorithm. The study [30] considered two kinds of cost, namely, the deployment cost and the area covered by edge servers, then a dynamic programming algorithm and the geometric image approach are used to find the placement solution. Another challenge of edge computing is that some servers may fail due to hardware faults or cyberattacks, whihc will greatly reduce the user experience. If users connected to the failed edge server cannot cover by any other edge servers, they need to access the service from remote cloud center. Therefore, Cui *et al.* [23], [24] considered the robustness of edge server network when deploying edge servers. Specifically, Cui *et al.* [24] proposed an k-edge server placement method by jointly considering user coverage and network robustness, Cui *et al.* [23] further formally formulated this robustness-oriented k-edge server placement problem and proposed an integer programming-based optimal approach. For optimizing energy consumption of edge servers, the study [22] proposed a particle swarm optimization-based energy-aware edge server placement algorithm, while LESP [31] proposed load-aware edge server placement method and designed tree-based placement strategy. In [18] and [19], a mixed integer programming-based approach is proposed to deploy edge server for making balance the workloads of edge servers and minimizing the access delay. For minimizing the distance between servers and WiFi access points, Lähderanta *et al.* [32] formulated the edge server placement as a capacitated location-allocation problem.

To support multiple applications in mobile-edge networks, Zhao and Liu [33] proposed latency-aware heuristic placement algorithm and SPAC [34] utilized local-search-based algorithm to minimize the weighted sum of the service cost and edge server opening. To add new servers to the edge network, the work [35] proposed a solution to the scale up the edge server deployment by selecting the optimal number of new edge servers and reallocate access points optimally to the old and new edge servers, and [36] improved the efficiency and reduced the cost of

edge provisioning by discovering proper unforeseen edge locations.

III. Problem Statement

In this section, we present the problem description of preference-aware edge server placement in IoT, which we prove is an NP-hard problem and formulate with multiobjective optimization. Some important mathematical symbols and their meanings are given in Table I.

In this article, the problem we need to solve is as follows. As Fig. 2 shows, in an area, a certain number of base stations and a small number of edge servers are discretely distributed. These base stations are responsible for receiving data and query requests from mobile users, which are transmitted to or processed by edge servers. If an edge server cannot deal with a request properly, it would be transmitted to the cloud server or other edge server for further processing. Assume that we already know the location information of all base stations, such as longitudes and latitudes, and need to find a deployment where positions of edge servers happen to coincide with the positions of some of base stations.

We can regard this problem as one related to networks. Given a complete graph $G = <V, E>$. V is the set of locations of all nodes, including B and S, which represents the set of base stations and edge servers, respectively. It is clear that $S \subset B$ because each server must be deployed where the base station is located. E is the set of edges in the graph, and there are transmission paths between every two base stations.

In fact, this problem can be reduced to a classic location problem, that is, the p-center problem. This problem describes a similar situation that in a weighted graph $G = <V, E>$, point set V represents the potential locations of service stations. We want to determine a deployment of service stations on this point set to minimize the maximum of length of paths from every node to the one in the control of it.

Lemma: The edge server placement problem is NP-Hard.

Proof: We can reduce the metric p-center problem to the edge server placement problem. Consider a metric p-center problem P with a weighted complete graph $G^* = <V^*, E^*>$, we can build an edge server placement problem P_1 with $G = <V, E>$ where $V = V^*$ and $E = E^*$. If we place p servers in G, the optimal solution is exactly the optimal solution to p-center problem with G^*. So P can be reduced to P_1, which can also be written as $P \leq P_1$. As we all know, the p-center problem is NP-hard, so the edge server placement problem is also NP-hard. ■

It is convenient to use the format of a multiobjective optimization problem to describe what we want to solve. Suppose that all base stations have the same workload while the number of query requests is different. The data set we use is the road traffic data set (see Section V-A). The data volume of each base station is set to the same. Generally speaking, as a 720P camera with the default code stream 3000 kb/s, 24 h of recording requires about 32-GB storage space, so the data volume is set to 32 GB. And the number of queries of the base station varies from a few times to hundreds of times (see Section I).

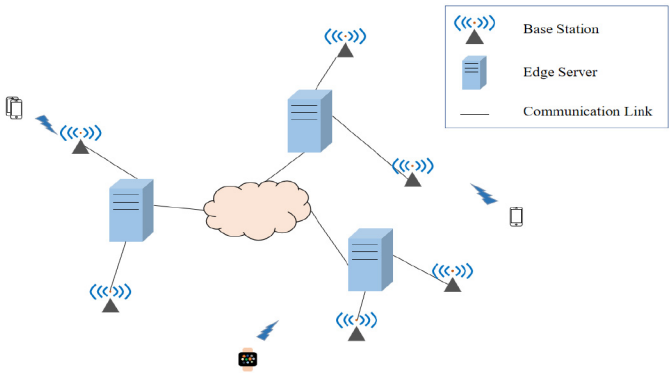

Fig. 2. Example of edge computing network.

TABLE I
NOTATIONS USED IN THIS ARTICLE

Symbols	Meaning
B	set of base stations
S	set of edge servers
N	the number of base stations
K	the number of edge servers
lb	the location of base station
ls	the location of edge server
M_i^R	the amount of data the i-th server receives
Q_i^R	the amount of query requests the i-th server receives
dis	a function to calculate the distance between two nodes

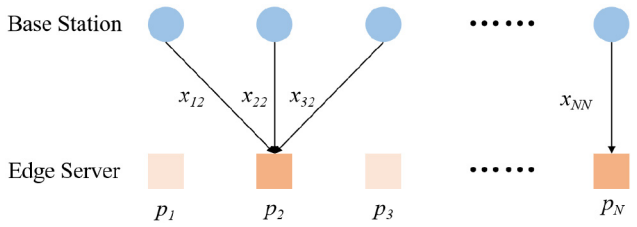

Fig. 3. Example of edge server deployment.

We need to find a matrix $\mathbf{X} = [x_{ij}]_{N\times N}$ and a vector $\mathbf{P} = [p_k]_{N\times 1}$, where

$$x_{ij} = \begin{cases} 1, & \text{if } B_i \text{ is in the control of } S_j \\ 0, & \text{else} \end{cases} \tag{1}$$

$$p_k = \begin{cases} 1, & \text{if an edge server is deployed on the location of } B_k \\ 0, & \text{else.} \end{cases} \tag{2}$$

For example, as Fig. 3 shows, we deploy edge servers at the location of the second, the fourth and the Nth base station. So p_2, p_4, and p_N are 1 while p_1, p_3 and other elements in $\mathbf{P}$ are 0. Arrows in the figure mean data transmission paths where the first, second and third base station give data to the second server and the Nth base station gives data to the Nth server. So x_{12}, x_{22}, x_{32}, and x_{NN} are 1 while x_{11}, x_{21} and so on are 0.

Our optimization goal consists of three parts. The first one is $M(\mathbf{X}, \mathbf{P})$ that represents memory balancing related to $\mathbf{X}$ and $\mathbf{P}$ expressed in (1) and (2). We want to minimize the maximum length of the interval consists of memory load of each server

to achieve balancing which can be expressed as

$$M(\mathbf{X},\mathbf{P}) = \max_{1\le i\le N} M_i^R - \min_{1\le j\le N} M_j^R = \max_{1\le i,j\le N}\left(M_i^R - M_j^R\right) \tag{3}$$

where i, j satisfy that $p_i, p_j = 1$, M_i^R means the amount of data S_i receives.

Similarly, the second optimization goal is $Q(\mathbf{X},\mathbf{P})$ that represents query balancing

$$Q(\mathbf{X},\mathbf{P}) = \max_{1\le i\le N} Q_i^R - \min_{1\le j\le N} Q_j^R = \max_{1\le i,j\le N}\left(Q_i^R - Q_j^R\right) \tag{4}$$

where i, j satisfy that $p_i, p_j = 1$, Q_i^R means the amount of query requests S_i receives.

The last one is $D(\mathbf{X},\mathbf{P})$ which is the sum of distances between base stations and edge servers and represents query latency

$$D(\mathbf{X},\mathbf{P}) = \sum_{1\le i,j\le N} \mathrm{dis}(lb_i, ls_j) \tag{5}$$

where i, j satisfy that $x_{ij} = 1$.

In (5), dis is a function to calculate the distance between two nodes. *lb* and *ls* means the location of base stations and edge servers, respectively. Next, we illustrate our optimization goals for edge server deployment with the following three constraints: 1) assuming that all the edge servers are placed, each base station must transmit its data and requests to one and only one server; 2) naturally, the number of servers is K; and 3) no data or requests should be given to other base stations where servers are not placed. As shown

$$\sum_{j=1}^{N} x_{ij} = 1, \sum_{k=1}^{N} p_k = K, \sum_{j=1}^{N} x_{ij}p_j = 1. \tag{6}$$

According to (1)–(6), our edge server deployment problem becomes a multiobjective problem following can be transformed into a single-objective problem following:

$$\begin{cases} \text{variable:} & \mathbf{X}, \mathbf{P} \\ \text{minimize:} & M(\mathbf{X},\mathbf{P}), Q(\mathbf{X},\mathbf{P}), D(\mathbf{X},\mathbf{P}) \\ \text{s.t.:} & \sum_{j=1}^{N} x_{ij} = 1 \\ & \sum_{k=1}^{N} p_k = K \\ & \sum_{j=1}^{N} x_{ij}p_j = 1 \\ & x_{ij}, p_k \in \{0,1\}. \end{cases} \tag{7}$$

As we can know, the format of the problem above can be transformed into a single-objective problem following with the same constraints:

$$\begin{cases} \text{variable:} & \mathbf{X}, \mathbf{P} \\ \text{minimize:} & \omega_1 M(\mathbf{X},\mathbf{P}) + \omega_2 Q(\mathbf{X},\mathbf{P}) + \omega_3 D(\mathbf{X},\mathbf{P}) \end{cases} \tag{8}$$

where $\omega_1, \omega_2, \omega_3 > 0$ and $\omega_1 + \omega_2 + \omega_3 = 1$.

Further, we can transform (7) into

$$\begin{cases} \text{variable:} & \mathbf{X}, \mathbf{P} \\ \text{minimize:} & \max\Big(\omega_1\left(M_i^R - M_j^R\right) + \omega_2\left(Q_i^R - Q_j^R\right) \\ & \qquad + \omega_3 \sum_{1\le i,j\le N} \mathrm{dis}(lb_i, ls_j)\Big). \end{cases} \tag{9}$$

Theorem 1: All the solutions of (8) are exactly the solutions of (7).

Proof: We can use contradiction to prove the theorem. If it is wrong, then there must exist $L = [\mathbf{X},\mathbf{P}]$, which is an optimal solution of (8) but not the optimal solution of question (7). Then there must be a solution $L^* = [\mathbf{X}^*,\mathbf{P}^*]$ that enables (7) reach the optimal. Therefore, $M(\mathbf{X}^*,\mathbf{P}^*) \le M(\mathbf{X},\mathbf{P})$, $Q(\mathbf{X}^*,\mathbf{P}^*) \le Q(\mathbf{X},\mathbf{P})$, $D(\mathbf{X}^*,\mathbf{P}^*) \le D(\mathbf{X},\mathbf{P})$. According to the definition, three parameters $\omega_1, \omega_2, \omega_3$ are larger than zero. It provides the evidence that

$$\begin{aligned} &\omega_1 M(\mathbf{X}^*,\mathbf{P}^*) + \omega_2 Q(\mathbf{X}^*,\mathbf{P}^*) + \omega_3 D(\mathbf{X}^*,\mathbf{P}^*) \\ &\le \omega_1 M(\mathbf{X},\mathbf{P}) + \omega_2 Q(\mathbf{X},\mathbf{P}) + \omega_3 D(\mathbf{X},\mathbf{P}). \end{aligned} \tag{10}$$

Equation (10) means the solution L^* is a more optimal solution of (8) and this fact is a contradiction of the assumption that L is optimal. ■

Theorem 2: All the solutions of (9) are exactly the solutions of (7).

Proof: What we need to do is to prove solution set $\mathbf{N}_1$ of (8) contains $\mathbf{N}_2$ of (9), that is, $\mathbf{N}_1 \subset \mathbf{N}_2$. If for every solution $L_1 = [\mathbf{X}_1, \mathbf{P}_1]$ which makes

$$\omega_1\left(M_i^R - M_j^R\right) + \omega_2\left(Q_i^R - Q_j^R\right) + \omega_3 \sum_{1\le i,j\le N} \mathrm{dis}\left(lb_i, ls_j\right)$$

reaches the maximum can also make $M(\mathbf{X},\mathbf{P})$, $Q(\mathbf{X},\mathbf{P})$ and $D(\mathbf{X},\mathbf{P})$ reach the maximum, the assumption above would be met. Similar to the proof process of Theorem 1, we can use contradiction to prove it. Therefore, $\mathbf{N}_1 \subset \mathbf{N}_2$ and all the solutions of (9) are exactly the solutions of (7). ■

IV. Approach

In this section we detail the proposed approaches, a QIP for small-scale data set and a heuristic algorithm (namely, TAKG) for large-scale data set.

A. Introduction to QIP

In this section, we formulate this problem to a QIP problem.

First, derive the expression of M_i^R and $M(\mathbf{X},\mathbf{P})$ from $\mathbf{X}$ and $\mathbf{P}$. We can create a matrix $\mathbf{M} = [M_{ij}]_{N\times N}$ that represents the potential amount of data transmitted from ith base station to edge server located at the position of jth base station. A property of this matrix is all the elements in a row are identical and equal to the amount of data the ith base station possesses. It can also be written as a row vector $\mathbf{M} = [\mathbf{M}_{*1}, \mathbf{M}_{*2}, \ldots, \mathbf{M}_{*N}]$, where column vectors $\mathbf{M}_{*k} (k = 1, 2, \ldots, N)$ are the same. For example, if the assumption that all the base stations have the same workload w is accepted, $\mathbf{M}$ will become a matrix with all the elements are w. $\mathbf{x}_{*k} (k = 1, 2, \ldots, N)$ is the kth column vector of matrix $\mathbf{X}$. M_i^R means the data that ith edge server receives and can be expressed as

$$M_i^R = p_i \sum_{z=1}^{N} M_{zi}x_{zi} = p_i \mathbf{M}_{*i}^T \mathbf{x}_{*i}. \tag{11}$$

So (3) becomes

$$M(\mathbf{X},\mathbf{P}) = \max_{1\le i,j\le N}\left(M_i^R - M_j^R\right) = \max_{1\le i,j\le N}\left(p_i\mathbf{M}_{*i}^T\mathbf{x}_{*i} - p_j\mathbf{M}_{*j}^T\mathbf{x}_{*j}\right) \quad (12)$$

where i, j satisfy that $p_i, p_j = 1$.

Second, $\mathbf{X}$ and $\mathbf{P}$ can be transformed into a vector $y = [\mathbf{x}_{*1}^T, \mathbf{x}_{*2}^T, \ldots, \mathbf{x}_{*N}^T, \mathbf{P}^T]^T$.

Third, in order to convert the problem into the format $y^T\mathbf{A}y$, $\mathbf{A}$ is also needed. Since the function max relates to the subscripts i and j when evaluating $M(\mathbf{X},\mathbf{P})$, each element of the coefficient matrix $\mathbf{A}$ should be related to the corresponding subscripts i and j. Let

$$\mathbf{A}_{ij}^{(1)} = \begin{bmatrix} \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{O} & \cdots & \mathbf{B}_i^{(1)} & \cdots & -\mathbf{B}_j^{(1)} & \cdots & \mathbf{O} \end{bmatrix}. \quad (13)$$

In the above equation, $\mathbf{O}$ means a zero matrix with the same size of $\mathbf{B}_i^{(1)}$ which is $N \times N$

$$\mathbf{B}_i^{(1)} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ M_{1i} & M_{2i} & \cdots & M_{Ni} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{M}_{*i}^T \\ \vdots \\ \mathbf{0} \end{bmatrix}. \quad (14)$$

Therefore, each element in $y^T\mathbf{A}y$ will be

$$y^T\mathbf{A}_{ij}^{(1)}y = \mathbf{P}^T\cdot\mathbf{B}_i^{(1)}\cdot\mathbf{x}_{*i} + \mathbf{P}^T\cdot\left(-\mathbf{B}_j^{(1)}\right)\cdot\mathbf{x}_{*j} = M_i^R - M_j^R. \quad (15)$$

So the optimization function of memory balancing is

$$\max_{1\le i,j\le N} y^T\mathbf{A}_{ij}^{(1)}y. \quad (16)$$

From (15) we can get N^2 matrixes but what we need is one part that i, j satisfy $p_i, p_j = 1$. So we choose K^2 elements and form a new matrix. Equation (16) can be simplified as

$$\max\left(Y^T\mathbf{A}^{(1)}Y\right) \quad (17)$$

where $\max(\mathbf{T})$ means a function to find the maximum element in the matrix $\mathbf{T}$. $\mathbf{A}^{(1)}$ is the first coefficient matrix and Y is a matrix where diagonal elements are all y

$$\mathbf{A}^{(1)} = \left[\mathbf{A}_{ij}^{(1)}\right]_{K\times K},\ Y^T = \begin{bmatrix} y^T & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & y^T \end{bmatrix}. \quad (18)$$

Similarly, we can create a matrix $\mathbf{Q} = [Q_{ij}]_{N\times N}$ and matrix $\mathbf{D} = [D_{ij}]_{N\times N}$, which represents the amount of data query request times and distances between every two base stations, respectively.

The optimization function of query balancing is

$$\max\left(Y^T\mathbf{A}^{(2)}Y\right) \quad (19)$$

where each element in $\mathbf{A}^{(2)}$ can be expressed similar to (13) and (14). The only difference is that $\mathbf{B}_i^{(2)}$ is related to different matrix $\mathbf{Q}$

$$\mathbf{A}_{ij}^{(2)} = \begin{bmatrix} \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} & \cdots & \mathbf{O} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{O} & \cdots & \mathbf{B}_i^{(2)} & \cdots & -\mathbf{B}_j^{(2)} & \cdots & \mathbf{O} \end{bmatrix} \quad (20)$$

$$\mathbf{B}_i^{(2)} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ Q_{1i} & Q_{2i} & \cdots & Q_{Ni} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{Q}_{*i}^T \\ \vdots \\ \mathbf{0} \end{bmatrix}. \quad (21)$$

And the optimization function of the sum of translation latency is

$$y^T\tilde{\mathbf{A}}^{(3)}y \quad (22)$$

$$\tilde{\mathbf{A}}^{(3)} = \begin{bmatrix} \mathbf{O} & \cdots & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{O} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{B}_1^{(3)} & \cdots & \mathbf{B}_N^{(3)} & \mathbf{O} \end{bmatrix}. \quad (23)$$

Since this objective function is a summation function, it differs from the above two functions in that the last line of $\tilde{\mathbf{A}}^{(3)}$ will contain N matrices

$$\mathbf{B}_i^{(3)} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ D_{1i} & D_{2i} & \cdots & D_{Ni} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{D}_{*i}^T \\ \vdots \\ \mathbf{0} \end{bmatrix}. \quad (24)$$

According to (13)–(24), the original optimization goal is $\max(Y^T\omega_1\mathbf{A}^{(1)}Y)$, $\max(Y^T\omega_2\mathbf{A}^{(2)}Y)$, $y^T\omega_3\tilde{\mathbf{A}}^{(3)}y$.

The third item can be added into the function max because if y is fixed, it is a constant. Here, for the sake of uniformity, we expand this objective function into a matrix

$$Y^T\mathbf{A}^{(3)}Y \quad (25)$$

where each element in $\mathbf{A}^{(3)}$ is the same

$$\mathbf{A}^{(3)} = \left[\tilde{\mathbf{A}}^{(3)}\right]_{K\times K}. \quad (26)$$

So according to (9), (25), and (26), the problem can be transformed into

$$\begin{cases} \text{variable:} & y \\ \text{minimize:} & \max\left(Y^T\left(\omega_1\mathbf{A}^{(1)} + \omega_2\mathbf{A}^{(2)} + \omega_3\mathbf{A}^{(3)}\right)Y\right) \\ \text{s.t.:} & \sum_{j=1}^N x_{ij} = 1 \\ & \sum_{k=1}^N p_k = K \\ & \sum_{j=1}^N x_{ij}p_j = 1 \\ & x_{ij}, p_k \in \{0,1\}. \end{cases} \quad (27)$$

Now the problem has become a QIP problem. Many papers have discussed the complexity of QIP. Due to the limitation of

the size of the paper and this is not our focus, we have adopted the conclusion of [37]: solving QIP is an NP-hard problem.

B. Introduction to TAKG

In this section we first propose an approximate approach, namely, TAKG, for solving large-scale edge server placement. Then we analyze the complexity of TAKG.

Solving a QIP problem is a very hard problem, and an accurate solution can usually be found only in scenarios of a small data set. When data set is large, the accuracy of the solution should be sacrificed to obtain an approximate solution in order to speed up the procedure. The heuristic algorithm has better performance widely known. Therefore, we adopt the idea of a heuristic algorithm named TABU, and designed an algorithm based on multiple iterations to solve it fast.

The innovation of our proposed algorithm is the combination of three classical algorithms: 1) TABU search; 2) *k*-means; and 3) genetic algorithm. The heuristic algorithm TABU search is the main part of the algorithm. We build a TABU table so that each adjustment of the solution is made in a better direction as possible, without revisiting the already visited solution. However, the initial solution to classical TABU search is generated randomly. We use *k*-means algorithm in generating the initial solution because *k*-means is good at solving the site selection problem, so that the initial solution will be much closer to the optimal solution than the random solution, which makes the rate of iterative convergence more rapidly and reduces the complexity. The three operators in the genetic algorithm, selection, crossover and mutation, constitute the operations of adjusting the current solution in each iteration. These three operators, whose principles come from biological inheritance in nature, are one of the core steps of genetic algorithms, and many practical applications have shown their powerful capabilities. Our algorithm combines the advantages of the three classical algorithms, and thus has result that far exceed those of a single algorithm for solving the edge server deployment problem.

The input of TAKG is a weighted complete graph, the weight of each point is divided into two parts, the amount of data and the number of queries. The location coordinates of each point has already been acknowledged. The output of the algorithm is a matrix $\mathbf{X}$ and a vector $\mathbf{P}$.

The main idea of the algorithm is as follows.

1) Use an existed algorithm to calculate the initial solution that satisfies the constraints, for example, *k*-means algorithm. Then calculate the value of the objective function (loss function).
2) When the solution obtained does not satisfy the judgment conditions, continue iterating. Judgment conditions here include judging whether the maximum number of iterations has been reached and whether the optimal solution has not been changed for a certain number of times.
3) In each iteration, some minor adjustments are made which is divided into two steps, a deployment step and an allocation step. Here, the minor adjustments include three variations in genetic algorithm: a) selection; b) crossover; and c) mutation. If the constraints are met, the loss function value of the new solution is calculated and the reference solution set is extended. In the deployment step we need to adjust the positions of the edge servers according to the vector $\mathbf{P}$. And in the allocation step we need to adjust how the base station transmits data based on the matrix $\mathbf{X}$. After a certain number of iterations, the optimal solution of these reference solution sets is found and compared with the previous one. Adding the optimal solution to the Tabu table at the same time.
4) If in the above process, a solution has already been in the Tabu table, give it up.

Algorithm 1 TAKG

```
Require: a weighted complete graph G
Ensure: X and P
  generate an initial solution solu using k-means or k-means++
  calculate the loss function value curr_quality
  initialize best_quality and best_solution
  while best_solution has not been unchanged for 20 times or not
  reach iter_num do
    clear candidates
    for i in range(100) do
      minimize the loss function value by genetic algorithm with
      respect to P
      minimize the loss function value by genetic algorithm with
      respect to X
      if satisfy the constraints then
        add a solution to candidates
      end if
    end for
    choose the minimum loss value curr_quality in candidates
    if curr_quality ⩽ best_quality then
      update best_quality and best_solution
    end if
    update Tabu_table
  end while
  return X and P
```

Next, the time complexity of TAKG will be analyzed. We can divide the algorithm into the following steps: generating the initial solution, making minor adjustments, judging the constraints, searching in the Tabu table and updating the Tabu table.

1) The complexity of generating the initial solution varies depending on the method we choose. With *k*-means algorithm, the time complexity becomes $O(N \cdot K)$.
2) The upper limit of complexity of making minor adjustments by the three operators in the genetic algorithm are equivalent to processing each element in P and X for at most a constant time. So it becomes $O(N + N^2)$.
3) It is obvious that the complexity of judging the constraints is $O(N + N^2)$.
4) The complexity of searching in the Tabu table and updating the Tabu table is related to the size of Tabu table. If we set the length of Tabu table is L, the minimum complexity will be $O(\log L)$.

In summary, after all the analysis, we can clearly obtain the time complexity of the TAKG algorithm as $O(N \cdot K) + O(M \cdot (N + N^2 + \log L))$, where M represents the maximum number of iterations.

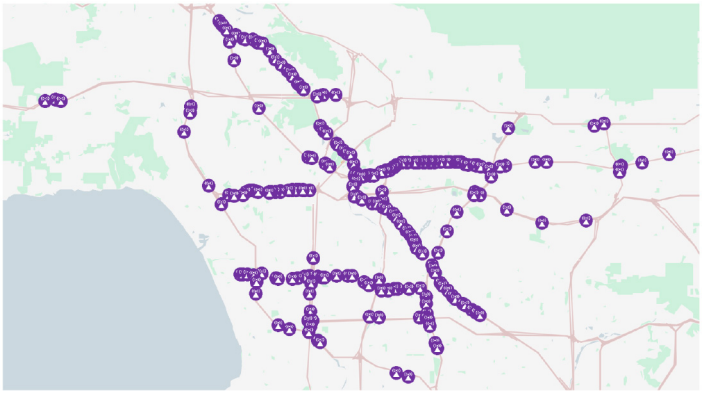

Fig. 4. Distribution of base stations.

V. Evaluation

In this section, we conduct an experimental evaluation of the proposed methods. By comparing memory balancing, query balancing and average access delay with other methods in the context of different number of base stations and different number of servers. Our experiments are divided into two parts, using QIP for exact solutions in small-scale case and heuristic algorithm for approximate solutions in large-scale case. We have determined that our methods perform more efficiently and effectively.

A. Data Set Discription

The data set employed in our experiments is from California's freeway, which contains traffic information collected from 1907 base stations along the road. The number of times the traffic department retrieves varies from location to location due to different traffic conditions as Fig. 4 shows. The amount of data stored in each device is assumed to be the same, while the number of queries will fluctuate widely from a few to several hundred, as can be seen in Fig. 1.

Since there are a large variety of IoT devices around us and the algorithm in this article has a better implementation for homogeneous devices, we choose the data set provided by collection devices with similar functionality. In the transportation system, the vehicle driving situation is the important information that the transportation department closely monitors. They may need to know the vehicle's path, whether it is a violation of the law, and other information to use as criteria for penalties for each driver.

B. Comparison Algorithm and Evaluation Metric

The comparison algorithms we use are as follows.

1) *K-Means:* The K-means clustering is the best known partitioning clustering algorithm, its simplicity and efficiency make it the most widely used of all clustering algorithms. Given a set of data points and the required number of clusters k, which is indicated by the user, the algorithm repeatedly divides the data into k clusters based on a distance function. This is an NP-hard problem but there exists efficient heuristic algorithms.
2) *PEMB:* PEMB (Place Edge Servers and Map Base Stations to Edge/Cloud Servers) is proposed by [38] and aims to optimize the average response time of the base stations which contain communication delay and task execution delay. It utilizes ILP solver. Integer linear programming is an approach to solve a linear objective function. In placement problem, when the optimization is a single objective, integer linear programming can achieve excellent results in terms of efficiency and accuracy.
3) *DCNOPA:* Divide and conquer-based near-optimal placement algorithm (DCNOPA) is proposed by [39] and considers the deployment cost and the traffic cost. It first selects k base stations with the largest coverage as the initial location. Then each cluster with a small number of clusters first selects the nearest base station within its average radius. The remaining base stations are allocated to the nearest cluster. Each cluster selects a server with the least average data traffic.
4) *MIQP:* Guo *et al.* [40] proposed an approximate approach that adopted the K-means and mixed-integer quadratic programming (MIQP) with objective to balance the workload between edge clouds and minimize the service communication delay of mobile users.

The metrics we adopt are as follows.

1) *Average Access Delay:* This can be expressed as

$$\tilde{D} = \frac{\sum_{1 \le i,j \le N} x_{ij} \mathrm{dis}\left(lb_i, ls_j\right)}{N}. \tag{28}$$

In our data set, the given location is the latitude and longitude of the base station, and through them the straight line distance between the base stations can be derived. Since our scenario is a wireless transmission scenario, the transmission delay can be represented by the average distance between each pair of base stations and edge servers.

2) *Memory Balancing:* We use standard deviation of memory load of each edge server to evaluate the effectiveness of memory balancing

$$\tilde{M} = \sqrt{\frac{\sum_{i=1}^{N}\left(M_i^R - \overline{M^R}\right)^2}{K}} \tag{29}$$

where M_i^R has been claimed before and $\overline{M^R}$ is the average of M_i^R.

3) *Query Balancing:* Similarly, the expression of query balancing is

$$\tilde{Q} = \sqrt{\frac{\sum_{i=1}^{N}\left(Q_i^R - \overline{Q^R}\right)^2}{K}} \tag{30}$$

where $\overline{Q^R}$ is the average of Q_i^R.

4) *Comprehensive Metric:* The three metrics we want to compare are normalized and weighted to obtain a comprehensive metric shown in the following equation:

$$\tilde{C} = \omega_1 \mathrm{nor}\left(\tilde{D}\right) + \omega_2 \mathrm{nor}\left(\tilde{M}\right) + \omega_3 \mathrm{nor}\left(\tilde{Q}\right) \tag{31}$$

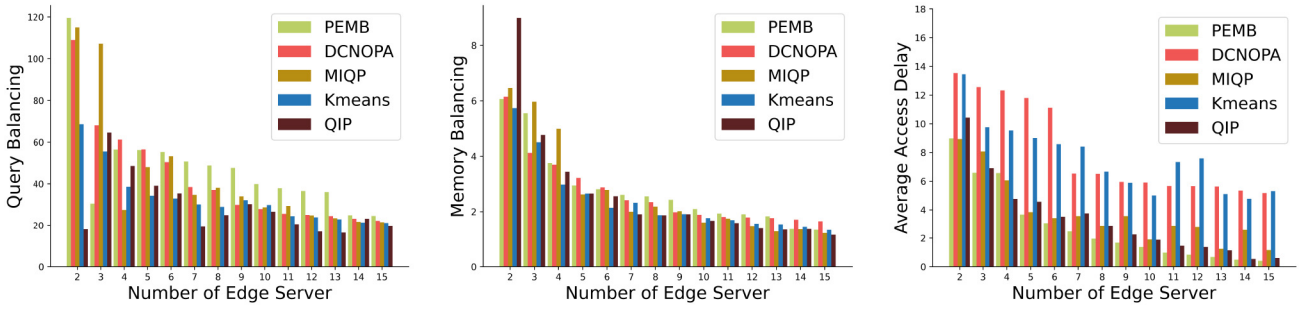

Fig. 5. Performance evaluation on 30 base stations. (a) Query balancing. (b) Memory balancing. (c) Average access delay.

TABLE II
COMPREHENSIVE METRIC ON SMALL-SCALE DATA SET WITH 30 STATIONS

Methods \ Number	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QIP	0.576	0.474	0.342	0.287	0.252	0.195	0.188	0.189	0.164	0.137	0.119	0.111	0.115	0.101
PEMB[38]	0.701	0.407	0.412	0.321	0.300	0.268	0.251	0.237	0.200	0.181	0.174	0.166	0.118	0.115
DCNOPA[39]	0.778	0.586	0.550	0.510	0.468	0.321	0.315	0.272	0.263	0.250	0.247	0.245	0.233	0.225
MIQP[40]	0.702	0.646	0.369	0.293	0.302	0.232	0.232	0.231	0.167	0.195	0.172	0.129	0.156	0.120
K-means	0.661	0.505	0.407	0.374	0.344	0.339	0.282	0.274	0.244	0.280	0.280	0.222	0.207	0.214

TABLE III
COMPREHENSIVE METRIC ON SMALL-SCALE DATA SET WITH 5 SERVERS

Methods \ Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29
QIP	0.229	0.248	0.243	0.255	0.271	0.293	0.359	0.381	0.358	0.376	0.400	0.421	0.485	0.516
PEMB[38]	0.336	0.496	0.483	0.335	0.367	0.349	0.372	0.400	0.429	0.459	0.458	0.486	0.515	0.548
DCNOPA[39]	0.560	0.413	0.454	0.550	0.404	0.410	0.423	0.409	0.482	0.481	0.475	0.613	0.610	0.653
MIQP[40]	0.285	0.355	0.306	0.318	0.347	0.380	0.383	0.418	0.412	0.422	0.418	0.451	0.474	0.625
K-means	0.490	0.477	0.426	0.592	0.486	0.480	0.434	0.490	0.422	0.439	0.497	0.507	0.529	0.544

where nor(x) is a normalization function. Here, we set ω_1, ω_2, and ω_3 are equal to 1/3 which will balance three aspects we need to observe.

C. Experiment on Small-Scale Data Set

We use a selected portion of the data set described in Section V-A to implement performance tests using different methods. First, we fix the number of base stations to 30 and change the number of edge server from 2 to 15 shown in Fig. 5 and Table II. Then, we set the number of edge server to 5 and change the number of base station from 16 to 29 in Fig. 6 and Table III where bold numbers indicate that this method performs best under this situation. From them, we observe the following.

1) When the number of base station is fixed, all the metrics go down when the number of edge server increases. Here, we can define a variable r, which represents the ratio between the number of edge server and the number of base station

$$r = \frac{K}{N}. \tag{32}$$

The larger r is, the less number of base stations each server has to control on average, and the easier it is for optimization. The most extreme case is when the number of server is equal to the number of base station, then servers can be deployed at each location of base station, and the average delay will become zero and the load balancing will become better.

2) When the number of edge server is fixed, there is an upward trend in all metrics as the number of base station increases. However, the change is small compared to Fig. 5. In Fig. 5, when the number of base stations is fixed, the value of r changes from 0.067 to 0.5, while in Fig. 6, the value of r changes from 0.167 to 0.333 where the range of r is smaller. Therefore, the magnitude of changes in metrics presented is also smaller.
3) QIP performs best in most cases. The comprehensive performance of PEMB is less than that of QIP when $k = 2$. This is due to the fact that the latency-optimal case considered by PEMB in this situation also happens to be the case where the query load balancing is better, but our multiple objectives have some mutual constraints.
4) As for the analysis of data, QIP performs 29.60%, 25.17%, 20.75%, and 10.60% better than the other three methods on average on 30 base stations and 21.34%, 38.99%, 37.50%, and 35.12% on five servers.
5) Overall, it appears that our method performs best on small data sets, followed by MIQP, then PEMB with similar DCNOPA and K-means performance. MIQP performs a little worse compared to QIP because it can get an approximate optimal solution on the basis of

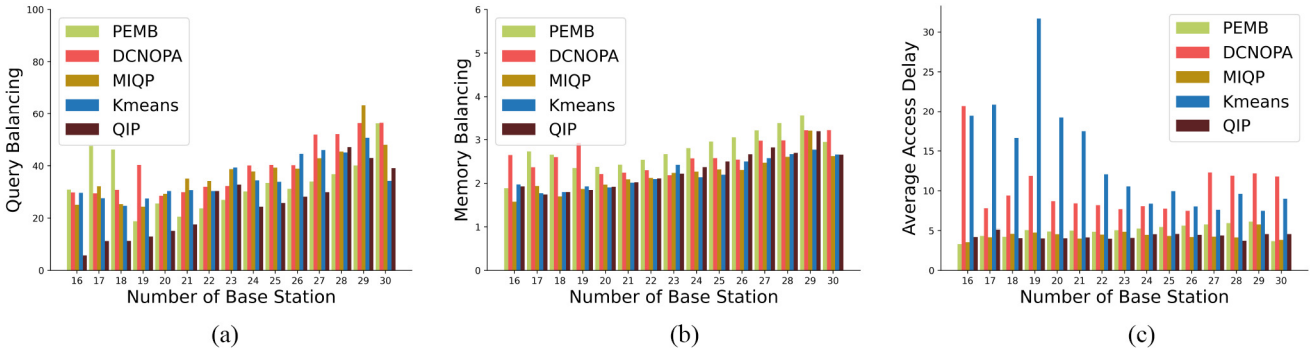

Fig. 6. Performance evaluation on 5 edge servers. (a) Query balancing. (b) Memory balancing. (c) Average access delay.

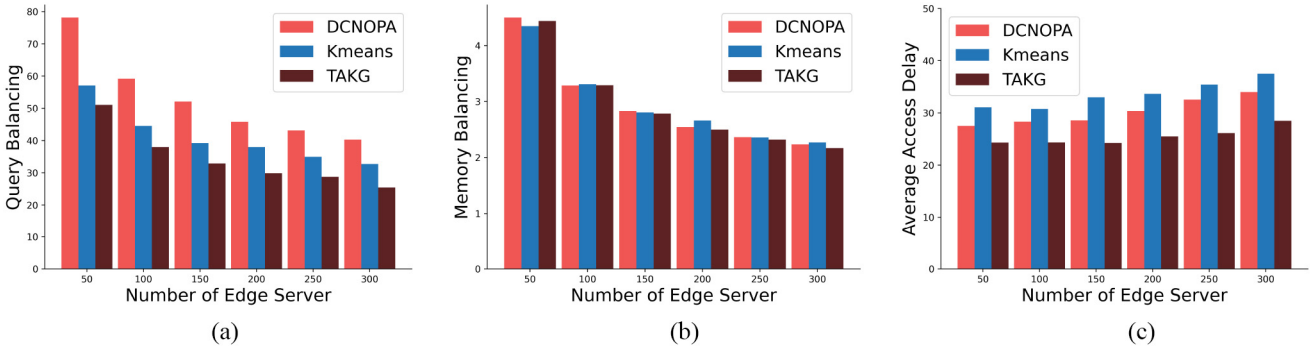

Fig. 7. Performance evaluation on 900 base stations. (a) Query balancing. (b) Memory balancing. (c) Average access delay.

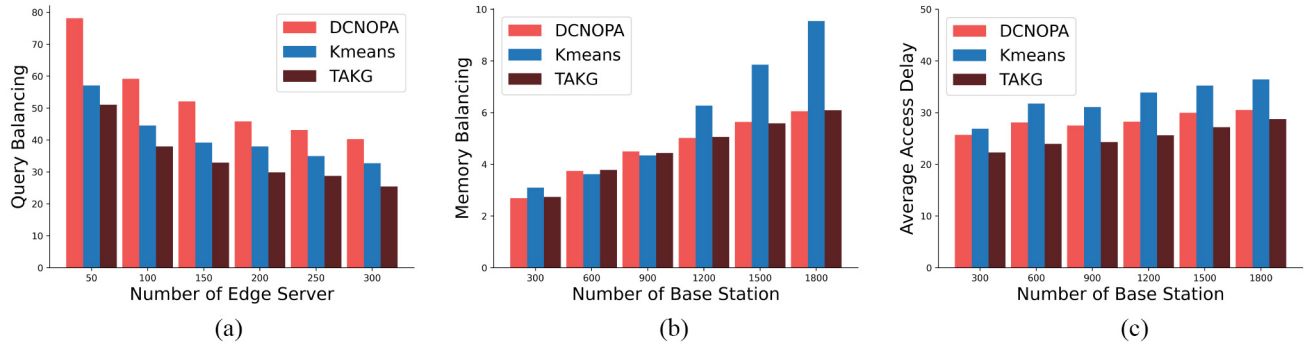

Fig. 8. Performance evaluation on 50 edge servers. (a) Query balancing. (b) Memory balancing. (c) Average access delay.

TABLE IV
COMPREHENSIVE METRIC ON LARGE-SCALE DATA SET WITH 50 SERVERS

Methods \ Number	300	600	900	1200	1500	1800
TAKG	0.686	0.559	0.505	0.485	0.474	0.470
DCNOPA[39]	0.757	0.637	0.601	0.592	0.574	0.576
K-means	0.820	0.672	0.616	0.587	0.583	0.575

K-means and MIQP but it neglects user preference, so query balancing is not that good.

D. Experiment on Large-Scale Data Set

We compare TAKG with *K*-means and DCNOPA which can also run fast on large data set. First, we set the number of edge server to be 50 and change the number of base station from 300 to 1800. As shown in Fig. 7 and Table IV. Then, we fix the number of base station to 900 and change the number of edge server from 50 to 300 in Fig. 8 and Table V. From them, we can observe the following.

1) When the number of edge servers is fixed, r becomes smaller as the number of base station increases, and all of these metrics have a tendency to increase. The reason for this phenomenon is similar to that described in the previous section, because as r increases, each server needs to receive less data from different base stations on average, which is more conducive to load balancing and latency optimization.
2) When the number of base station is fixed, r becomes larger as the number of server increases, but the comprehensive metric does not have a tendency to decrease. The reason for this phenomenon is as r increases to a

TABLE V
COMPREHENSIVE METRIC ON LARGE-SCALE DATA SET WITH 900 STATIONS

Methods \ Number	50	100	150	200	250	300
TAKG	0.357	0.438	0.471	0.525	0.553	0.601
DCNOPA[39]	0.423	0.519	0.569	0.624	0.679	0.716
K-means	0.440	0.510	0.539	0.709	0.812	0.900

threshold, continuing to increase r will not produce a better performance optimization. For example, when the number of base stations is fixed at 900, it does not make much difference whether the number of servers is 899 or 898, when the value of r is already very large.

3) After calculating, TAKG performs 34.04% and 27.97% better than K-means and DCNOPA, respectively, on 50 edge servers and 35.37% and 17.12% on 900 base stations.
4) In general, TAKG outperforms K-means and DCNOPA, as shown in Tables IV and V, TAKG is the optimal kind of approach in most cases. This is because although TAKG is worse than DCNOPA in the optimization of the average delay because this is its main optimization, TAKG does better than DCNOPA in the two evaluation parts of load balancing.
5) The variation in the average delay is smaller than load balancing, which may be due to the fact that our base station distribution is concentrated and dense. When r changes, the location of the selected server does not change much or may not even change compared to the previous location, resulting in small fluctuations in the distance that each base station transmits data.

VI. CONCLUSION

In this study, we present a preference-aware edge server placement method in IoT environment to provide better workload distribution by minimizing query latency and balancing the load of edge servers. To the best of our knowledge, this is the first work on preference-aware edge server placement in IoT. First, we formulated the preference-aware edge server placement with three optimization goals, including memory balancing, query balancing and average access delay. Then, we modelled it in the form of QIP problem with NP-hard complexity. Finally, we designed two approaches to solve this problem, namely, a QIP for small-scale data set and a heuristic algorithm named TAKG for large-scale data set. We conduct extensive experiments over a large real-world data set, and the experimental results show that the proposed methods beat all baselines in terms of query latency and load balancing.

In future work, we plan to improve edge server placement in IoT environment by considering online placement edge servers to dynamically update the deployment of services. Additionally, in practice, edge servers can be heterogeneous due to different application scenarios, device brands, device types, and so on, which will make the device's capacity of computation and storage or energy consumption vary greatly. With the increasingly deep relationship between mobile devices and users, certain characteristics of users will be displayed on them, such as mobility and sociality. Therefore, taking into account the heterogeneity, mobility and sociality is also a future direction.

REFERENCES

[1] Z. Zheng *et al.*, "A fused method of machine learning and dynamic time warping for road anomalies detection," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 26, 2020, doi: 10.1109/TITS.2020.3016288.

[2] T. Wang, H. Luo, X. Zeng, Z. Yu, A. Liu, and A. K. Sangaiah, "Mobility based trust evaluation for heterogeneous electric vehicles network in smart cities," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1797–1806, Mar. 2021.

[3] Y. Chen, M. Zhou, Z. Zheng, and D. Chen, "Time-aware smart object recommendation in social Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2014–2027, Mar. 2020.

[4] Y. Chen, J. Zhang, M. Guo, and J. Cao, "Learning user preference from heterogeneous information for store-type recommendation," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1100–1114, Nov./Dec. 2020.

[5] Y. Wang, J. Shen, and Y. Zheng, "Push the limit of acoustic gesture recognition," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2020, pp. 566–575.

[6] J. Zhang, S. Zhong, T. Wang, H.-C. Chao, and J. Wang, "Blockchain-based systems and applications: A survey," *J. Internet Technol.*, vol. 21, no. 1, pp. 1–14, 2020.

[7] J. Zhang, S. Zhong, J. Wang, X. Yu, and A. Osama, "A storage optimization scheme for blockchain transaction databases," *Comput. Syst. Sci. Eng.*, vol. 36, no. 3, pp. 521–535, 2021.

[8] J. Shen, J. Cao, and X. Liu, "BaG: Behavior-aware group detection in crowded urban spaces using WiFi probes," *IEEE Trans. Mobile Comput.*, early access, Jun. 2, 2020, doi: 10.1109/TMC.2020.2999491.

[9] J. Shen, J. Cao, X. Liu, and S. Tang, "SNOW: Detecting shopping groups using WiFi," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3908–3917, Oct. 2018.

[10] X. Liu, P. Lin, T. Liu, T. Wang, A. Liu, and W. Xu, "Objective-variable tour planning for mobile data collection in partitioned sensor networks," *IEEE Trans. Mobile Comput.*, early access, Jun. 17, 2020, doi: 10.1109/TMC.2020.3003004.

[11] T. Wang *et al.*, "Privacy-enhanced data collection based on deep learning for Internet of Vehicles," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6663–6672, Oct. 2020.

[12] Y. Chen, J. Zhang, L. Xu, M. Guo, and J. Cao, "Modeling latent relation to boost things categorization service," *IEEE Trans. Services Comput.*, vol. 13, no. 5, pp. 915–929, Sep./Oct. 2020.

[13] X. Liu, M. S. Obaidat, C. Lin, T. Wang, and A. Liu, "Movement-based solutions to energy limitation in wireless sensor networks: State of the art and future trends," *IEEE Netw.*, vol. 35, no. 2, pp. 188–193, Mar./Apr. 2021.

[14] T. Wang, L. Qiu, A. K. Sangaiah, A. Liu, M. Z. A. Bhuiyan, and Y. Ma, "Edge-computing-based trustworthy data collection model in the Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4218–4227, May 2020.

[15] T. Wang, H. Ke, X. Zheng, K. Wang, A. K. Sangaiah, and A. Liu, "Big data cleaning based on mobile edge computing in industrial sensor-cloud," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1321–1329, Feb. 2020.

[16] H. Jayakumar, A. Raha, and V. Raghunathan, "Energy-aware memory mapping for hybrid FRAM-SRAM MCUS in IoT edge devices," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embedded Syst. (VLSID)*, 2016, pp. 264–269.

[17] M. Singhal *et al.*, "Collaboration in multicloud computing environments: Framework and security issues," *Computer*, vol. 46, no. 2, pp. 76–84, 2013.

[18] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, May 2019.

[19] S. Kasi, M. Kasi, K. Ali, M. Raza, and A. Lasebae, "Heuristic edge server placement in industrial Internet of Things and cellular networks," *IEEE Internet Things J.*, early access, Dec. 1, 2020, doi: 10.1109/JIOT.2020.3041805.

[20] 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, 2019.

[21] Y. Ren, F. Zeng, W. Li, and L. Meng, "A low-cost edge server placement strategy in wireless metropolitan area networks," in *Proc. IEEE 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2018, pp. 1–6.

[22] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, 2018, pp. 66–73.
[23] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, "Robustness-oriented *k*-edge server placement," in *Proc. 20th IEEE/ACM Int. Symp. Clust. Cloud Internet Comput. (CCGRID)*, 2020, pp. 81–90.
[24] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, "Trading off between user coverage and network robustness for edge server placement," *IEEE Trans. Cloud Comput.*, early access, Jul. 10, 2020, doi: 10.1109/TCC.2020.3008440.
[25] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[26] Y. Qian, R. Wang, J. Wu, B. Tan, and H. Ren, "Reinforcement learning-based optimal computing and caching in mobile edge network," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2343–2355, Oct. 2020.
[27] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. 39th IEEE Conf. Comput. Commun. (INFOCOM)*, Toronto, ON, Canada, Jul. 2020, pp. 2076–2085.
[28] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5853–5863, Jun. 2019.
[29] D. Bhatta and L. Mashayekhy, "Generalized cost-aware cloudlet placement for vehicular edge computing systems," in *Proc. CloudCom*, 2019, pp. 159–166.
[30] F. Wang, X. Huang, H. Nian, Q. He, Y. Yang, and C. Zhang, "Cost-effective edge server placement in edge computing," in *Proc. 5th Int. Conf. Syst. Control Commun.*, 2019, pp. 6–10.
[31] X. Xu *et al.*, "Load-aware edge server placement for mobile edge computing in 5G networks," in *Proc. Int. Conf. Service Orient. Comput.*, 2019, pp. 494–507.
[32] T. Lähderanta *et al.*, "Edge server placement with capacitated location allocation," 2019. [Online]. Available: arXiv:1907.07349.
[33] L. Zhao and J. Liu, "Optimal placement of virtual machines for supporting multiple applications in mobile edge networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6533–6545, Jul. 2018.
[34] J. Meng *et al.*, "Joint heterogeneous server placement and application configuration in edge computing," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2019, pp. 488–497.
[35] L. Lovén *et al.*, "Scaling up an edge server deployment," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, 2020, pp. 1–7.
[36] H. Yin *et al.*, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1031–1045, Apr. 2017.
[37] P. M. Pardalos and S. Jha, "Complexity of uniqueness and local search in quadratic 0–1 programming," *Oper. Res. Lett.*, vol. 11, no. 2, pp. 119–123, 1992.
[38] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 494–503, Jan. 2021.
[39] L. Zhao, J. Liu, Y. Shi, W. Sun, and H. Guo, "Optimal placement of virtual machines in mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2017, pp. 1–6.
[40] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C.-H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Softw. Pract. Exp.*, vol. 50, no. 5, pp. 489–502, 2020.

Yuanyi Chen received the B.Sc. degree from Sichuan University, Chengdu, China, in 2010, the M.Sc. degree from Zhejiang University, Hangzhou, China, in 2013, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2017.

He is currently a Distinguished Research Fellow with the Department of Computer Science and Computing, Zhejiang University City College, Hangzhou. He has published more than 20 technical papers in major international journals and conference proceedings. His research interest includes the Internet of Things, mobile computing, and ubiquitous computing.

Yihao Lin received the B.S. degree from the School of Mathematical Sciences, Zhejiang University, Hangzhou, China, in 2020. He is currently pursuing the master's degree with the College of Computer Science and Technology, Zhejiang University and the Zhejiang University City College, Hangzhou.

His research topic includes the Internet of Things and mobile-edge computing.

Zengwei Zheng received the B.S. and M.Ec. degrees in computer science and western economics from Hangzhou University, Hangzhou, China, in 1991 and 1998, respectively, and the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, in 2005.

He is currently a Professor with the Department of Computer Science and Engineering, Zhejiang University City College, Hangzhou. His research interests include wireless sensor network, location-based service, Internet of Things, digital agriculture and pervasive computing.

Prof. Zheng is a member of the ACM and the CCF.

Peng Yu received the B.S. degree in computer science from Soochow University, Suzhou, China, in 2020. He is currently pursuing the master's with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, and the Zhejiang University City College, Hangzhou.

His research topic includes the Internet of Things and mobile-edge computing.

Jiaxing Shen received the B.E. degree in software engineering from Jilin University, Changchun, China, in 2014, and the Ph.D. degree in computer science from PolyU, Hong Kong, in 2019.

He is currently a Research Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He has published several papers in high-impact journals and top conferences. His research interests include mobile computing, data mining, social computing, affective computing, and Internet of Things.

Dr. Shen served as a reviewer for many international conferences and journals like IEEE TRANSACTIONS ON MOBILE COMPUTING, IJCAI, and PERCOM.

Minyi Guo (Fellow, IEEE) received the B.Sc. and M.E. degrees in computer science from Nanjing University, Nanjing, China, in 1982 and 1986, respectively, and the Ph.D. degree in computer science from the University of Tsukuba, Tsukuba, Japan, in 1998.

He is currently a Zhiyuan Chair Professor and the Chair with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.

Dr. Guo received the National Science Fund for Distinguished Young Scholars Award from NSFC in 2007.