Building Mobile Edge Infrastructure in an Uncertain Environment

Xiao Chen
Department of Computer Science
Texas State University
San Marcos, TX 78666
xc10@txstate.edu

***Abstract*—With the prevalence of mobile devices, more and more mobile applications are offloading computation-intensive jobs to remote cloud data centers. Although such operations could substantially enhance the capability of mobile devices, a long communication delay is inevitable. To mitigate the problem, mobile edge computing (MEC) has emerged, empowering cloud networks by pushing resources to the network edge to be near end-users. There are two edge providers: edge infrastructure providers (EIPs) and edge service providers (ESPs). Most work on MEC so far has fallen into the arena of ESPs, who are thinking about how to aggregate resources from multiple EIPs to offer value-added services to end-users. In this paper, however, we discuss the problems faced by EIPs, because only when EIPs install sufficient edge resources in the right places will the ESPs be able to provide better services to the end-users. More specifically, we ask: where should we install edge resources and how many should we install, based on user movement and demand? To answer the first question, we use the clustering and elbow methods, and for the second question, we formulate an optimization problem to minimize the expected cost of over-installation and under-installation. To test the feasibility of our approaches, we implement the solutions using a real trace dataset of taxi cabs in San Francisco as an example. We hope our methods can guide the EIPs to build a good edge infrastructure so that the ESPs can further improve user experience on a solid physical foundation.**

***Index Terms*—area of interest, cloud, edge infrastructure providers, edge resources, edge service providers**

I. Introduction

With the prevalence of mobile devices, such as smartphones and tablet computers, more and more mobile applications are offloading computation-intensive jobs to remote cloud data centers [10]. Although such operations could substantially enhance the capability of mobile devices, a long communication delay is inevitable. To mitigate the problem, *mobile edge computing* (MEC) has emerged and empowered cloud networks by pushing resource, e.g., storage and processing capacity, to the network edge to be close to the end-users, thus providing users with quick and powerful computing, energy efficiency, storage capacity, and mobility support [8].

Fig. 1 shows an edge-cloud network consisting of a cloud, edge resources, and mobile end-users. The edge resources are located in various places and serve a large number of end-users. The users can access the edge resources through access points (APs). The MEC technology reduces the load on the core network, decreases communication delay, and hosts applications and services in a more cost-effective manner.

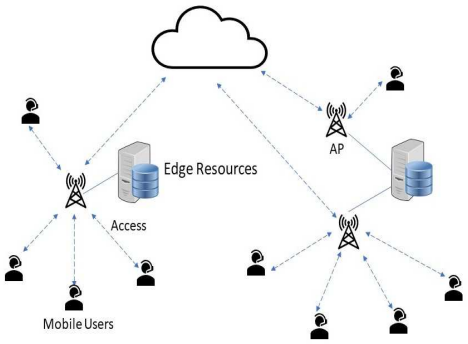

Fig. 1. Edge resources and mobile end-users in an edge-cloud network

There are two edge providers [13]: edge infrastructure providers (EIPs), who manage the physical edge infrastructure, and edge service providers (ESPs), who aggregate resources from multiple EIPs to place service entities and offer value-added services to end-users. Most work on MEC thus far has fallen into the arena of ESPs. These papers assume that the edge resources are deployed and then design algorithms to address problems such as *computation offloading* [6], [13], [16], *low latency* [3], [5], [11], *storage* [8], and *energy efficiency* [4], [7]. Computation offloading is a process of migrating computing tasks from mobile devices to more powerful sources, such as clouds, grids, or clusters [12]. This is an essential component of MEC because mobile devices have limited resources that are not adequate to handle computation-intensive tasks. For low latency, it is one of the motivations for building MEC: to reduce the latency between mobile devices and clouds to improve user experience. The storage aspect is straightforward because end-users need to utilize edge resources to overcome their device storage limitations. And energy efficiency is, in most cases, addressed indirectly via computational offloading.

In this paper, unlike most of the discussions in the literature, we investigate problems for the EIPs. We plan to improve MEC efficiency from the very beginning when EIPs deploy physical edge resources. They need to make the following decisions: where to install the edge resources and how many edge resources to place in a location to satisfy end-users' demand.

These questions are challenging because we are building edge infrastructure in an uncertain environment where users move around. Moreover, these questions are very important, as if sufficient resources are placed in the right locations based on user mobility and demand, it will facilitate the algorithms of offloading, latency, storage, and energy as the resources are available where users need them the most. Furthermore, by answering these two questions, EIPs are able to calculate their investment in the business and get the most profit. Therefore, in this paper, we propose feasible solutions to these practical questions based on statistical data. To answer the first question, we use clustering [18] and elbow [17] methods to identify *Areas of Interest* (AoIs) where we can install edge resources. For the second question, in each AoI, we formulate the question as a minimization problem to find the best number of resources to install to minimize the expected cost of the over-installation and under-installation. We give a detailed description of these methods and implement the solutions using a real trace dataset of taxi cabs in San Francisco [15] to test our methods. Implementation results show that our methods are feasible and can provide some guidance to EIPs to build MEC in an uncertain environment.

The differences of our work from others and the key contributions of our work are as follows:

- we consider the practical problems faced by EIPs in building MEC.
- we propose methods to solve the problems in an uncertainty environment.
- we implement our solutions using real trace data, verifying the feasibility of our methods.

The rest of the paper is organized as follows: Section II references the related work. Sections III and IV present the problems and their solutions. Section V describes the implementation of the methods, and the conclusion is in Section VI.

II. Related Work

Since the emergence of MEC, research in the literature has mostly focused on the following aspects.

Offloading. In [16], Sardellitti et al. proposed an algorithm-based design, called successive convex approximation (SCA), which optimizes computational offloading across densely deployed multiple radio access points. Chen et al. [6] designed an efficient computation offloading model using a game theoretic approach in a distributed manner. The game theory targeted the NP-hard problem of computation offloading incurred by multiuser computation offloading and provided a solution by attaining the Nash equilibrium of the multiuser computation offloading game. In [13], Ma et al. aimed to optimize the revenue earned by each edge node by optimally offloading tasks to the edge nodes. The authors formulated the NP-hard revenue-driven online task offloading (ROTO) problem and put forward the Level Balanced Allocation (LBA) algorithm to solve it.

Low latency. In [3], Abdelwahab et al. proposed REPLISOM, an edge cloud architecture and LTE enhanced memory replication protocol, to avoid latency issues. The LTE-integrated edge cloud provides computation and storage resources at the edge for resource-intensive services. Kumar et al. [11] presented a vehicular delay-tolerant network (VDTN)-based smart grid data management scheme. The authors investigated the use of VDTNs to transmit data to multiple smart grid devices, exploring the MEC environment. A store-and-carry forward mechanism for message transmission was used to avoid possible network bottlenecks and data latency. In [5], Chen et al. considered a general edge scenario of bursty requests, and put forward DeepLoad, an intelligent scheduler for offloading bursty requests to collaborating edge servers via deep reinforcement learning in edge environments. Their goal was to maximize the number of requests whose completion times were ahead of their deadlines.

Storage. In [8], Jararweh et al. presented a software-defined system for MEC (SDMEC). The framework connected software-defined system components to MEC to further extend its capabilities.

Energy efficiency. In [4], Beck et al. proposed ME-VoLTE, an architecture that integrates MEC with voice over LTE. The encoding of video calls is offloaded to the MEC server located at the base station (eNodeB). Offloading video encoding through external services helps to increase the battery lifetime of the user equipment and enhances power management. El-Barbary et al. [7] put forward DroidCloudlet, which is based on commodity mobile devices and can enhance mobile battery lifetime by migrating data-intensive and compute-intensive tasks to rich media.

All of the above research aims to improve user experience after edge resources are deployed. We will discuss problems that arise when people decide to install these resources in an edge infrastructure. We believe that if a sufficient number of edge resources are placed in the right locations where mobile users need them the most initially, it will facilitate solutions to the aforementioned problems and further improve user satisfaction. This is the task addressed in this paper.

III. Where to install resources?

In this section, we will address the first question: where to install edge resources. To answer the question, we need to find the Areas of Interest (AoIs) based on user mobility. Intuitively, an AoI is an area that many users go to frequently. Thus, we can pick several observation periods in the user mobility data and extract user locations in each period. Then, we apply clustering analysis [1] to the user locations (data points). The clusters identified are the AoIs in that observation period. We repeat the process for all the observation periods and the final AoIs are the overlap of these AoIs. Once an AoI is known, we can install edge resources in a central location such as coffee shop, station, or intersection in the AoI.

In the above method, when we do the clustering, a challenge is to find an appropriate clustering algorithm and a proper cluster number since user locations change all the time. To address these issues, we adopt the *Hierarchical Agglomerative Clustering* (HAC) method [14] and the *Elbow* method [17] to

generate clusters and decide a suitable cluster number. The algorithm *finding areas of interest* (FAI) that embeds the two methods is described in Algorithm 1.

Algorithm 1 Finding Areas of Interest (FAI)

Require: Input: m user locations as data points
Output: Clusters C for each k ($1 \leq k \leq m$), vector J containing all the J values, a suitable cluster number k

1: /* Clustering for each k */
2: put each data point into a cluster;
3: $k = m$; $J = \emptyset$;
4: **while** $k > 1$ **do**
5: combine the two clusters with the smallest distance into one and treat it as one cluster in the next round;
6: calculate $J_k = \sum_{j=1}^{k} \sum_{i \in C_j} ||d_i - o_j||_2^2$ and append it to vector J;
7: $k = k - 1$;
8: **end while**
9: /* determine a suitable k using the elbow method */
10: reverse order the elements in J;
11: draw the relationship between J and corresponding k;
12: identify the elbow and decide an appropriate k;

In the algorithm, we input m user locations as data points for clustering. The maximum number of clusters for m data points is m (each data point is a cluster) and the minimum number of clusters is one (all the data points are put into one cluster). To decide a suitable cluster number, we output the clusters C for each k in $[1, m]$. We use a bottom-up approach. In step 2 of the algorithm, we put each data point into a cluster. The number of clusters k is m at this point. We initialize a vector J, which will hold J values later, to be empty. Then we go into a loop. As long as the number of clusters k is greater than one, we combine two clusters with the smallest distance into one. One can use many metrics to measure the distance between the two clusters. For example, we can use the distance between two closest points in the two clusters, the distance between two farthest points in the two clusters, and so on. Here we use a popular distance metric: the distance between the centroids of the two clusters. Next, we calculate the J value of the current clusters in step 6 and append it to the J vector. In the calculation of the J value, we sum up in all the clusters the distance squared between each data point d_i in a cluster and its centroid o_j. Now, the two closest clusters are combined into one, and the number of clusters k is reduced by one. This process continues until all the clusters are grouped into one.

At this point, the loop is terminated and the J values of all the ks are stored into vector J. Next, we can plot the relationship between k and J in a graph to determine an appropriate k using the elbow method. So far, the J values added in each round have been in increasing order. As the number of clusters k decreases in each round, the J value increases, as is proved in Theorem 1. In order to show the elbow naturally, we reverse-order the elements in J in step 10. After we reorder them, they will be in decreasing order. When we plot the figure, an elbow will appear and the values on the elbow are the candidates for k. In mathematical optimization, the elbow is a common heuristic cutoff point where diminishing returns are no longer worth the additional cost. In this clustering algorithm, this means one should stop when combining another cluster doesn't give much better modeling of the data. Since the elbow method is a heuristic method, the candidates at the elbow may not be unique. Several points can be good choices as long as they have a clear explanation.

Theorem 1: In Algorithm 1, the J value calculated in each round of the loop satisfies: $J_k < J_{k-1}$, for $k = m, m-1, \cdots, 2$.

Sketch of proof. J_k is the J value when there are k clusters and after two closest clusters are combined into one, the J value of the resultant clusters is J_{k-1}. So, the difference between J_k and J_{k-1} lies in the two clusters that are combined. Assume these two clusters are C_i and C_j with centroids o_i and o_j, respectively. The combined cluster is C_{ij} with a centroid of o_{ij}. For a data point d_i that belongs to C_i, we have $||d_i - o_{ij}||_2^2 > ||d_i - o_i||_2^2$. Likewise, for a data point d_j that belongs to C_j, we have $||d_j - o_{ij}||_2^2 > ||d_j - o_j||_2^2$. That is, the distance between any data point in these two clusters and the new centroid o_{ij} is larger than the distance between the data point and its original centroid before the merging. The reason is as follows: let us look at cluster C_i and suppose d_j is the closest data point in C_j to any data point in C_i. That is, d_j makes $||d_j - d_i||_2^2$ minimum among all the data points in C_j and C_i. Now, for any two data points d_i and d_i' in C_i, we have $||d_i - d_j||_2^2 > ||d_i - d_i'||_2^2$. Otherwise, d_j should have been in the same cluster as one of them before d_i and d_i' were put into one cluster. If cluster C_i has only one data point, then d_i and d_i' are the same data point and the result is obvious. Therefore, the joining of d_j into cluster C_i makes the distance squared from each data point in cluster C_i to the new centroid larger than that to the original centroid o_i. Note that d_j is the closest data point to cluster C_i. If we add other farther away data points in C_j to cluster C_i, the conclusion will be even more so. Similarly, if we look at cluster C_j and add data points from C_i to C_j, the distance squared between each data point in cluster C_j and the new centroid will also be increased. Since all the data points in C_i and C_j have their distances to the new centroid increased from before, $J_k < J_{k-1}$. That proves the theorem. □

IV. How many resources to install?

Now that we have decided the AoIs for the edge resources, the next question is how many edge resources to install in each location. The answer to this question depends on user demand, which we assume is proportional to the number of users in the AoI. Intuitively, the more users go to the AoI, the greater the demand. We formulate this problem specifically below.

A. Problem Formulation

Suppose we have identified an AoI in which we plan to install edge resources. The user demand for these resources

in the AoI is not known due to the mobility of the users. If we install more resources than demand (over-deployment), there is a penalty denoted by O_p. If we install fewer resources than needed (under-deployment), there is a penalty denoted by U_p. For each resource, let P_c be the purchase cost, and R_u be the revenue brought in by the resource. The unit of over-allocation penalty $O_p = P_c$, and the unit of under-allocation penalty $U_p = R_u - P_c$.

The demand X (in the unit of the number of resources) at the location is unknown and assumed to be a continuous random variable. We denote $f(x)$ as the PDF of X and $F(x)$ as the CDF of X. We need to decide the number of resources u to install in order to

$$\begin{aligned} &\min_{u} z = E\left[O_p \cdot \max(0, u - X) + U_p \cdot \max(0, X - u)\right] \\ &\text{subject to } u \geq 0 \end{aligned} \tag{1}$$

In (1), $E[\cdots]$ is the expected value. Our goal is to find the optimal number of resources u so that the expected total cost of over-deployment and under-deployment can be minimized.

B. Our Solution

To solve the minimization problem in (1), we take the following steps. We first expand the expected value, z.

$$\begin{aligned} z &= \int_{x=0}^{\infty} \left[O_p \max(0, u - x) + U_p \max(0, x - u)\right] f(x)\, dx \\ &= \int_{x=0}^{u} \left[O_p(u - x)\right] f(x)\, dx + \int_{x=u}^{\infty} \left[U_p(x - u)\right] f(x)\, dx \\ &= O_p \int_{x=0}^{u} (u - x) f(x)\, dx + U_p \int_{x=u}^{\infty} (x - u) f(x)\, dx \\ &= O_p \left[u \int_{x=0}^{u} f(x)\, dx - \int_{x=0}^{u} x f(x)\, dx\right] + \\ &\quad U_p \left[\int_{x=u}^{\infty} x f(x)\, dx - u \int_{x=u}^{\infty} f(x)\, dx\right] \\ &= O_p \left[uF(u) - \int_{x=0}^{u} x f(x)\, dx\right] + \\ &\quad U_p \left[\int_{x=u}^{\infty} x f(x)\, dx - u(1 - F(u))\right] \end{aligned} \tag{2}$$

Now we show that z is a convex function. We take the first derivative

$$\begin{aligned} \frac{dz}{du} &= O_p \left[F(u) + uf(u) - uf(u)\right] + U_p \left[-uf(u) - \right. \\ &\qquad \left. (1 - F(u)) + uf(u)\right] \\ &= O_p F(u) - U_p(1 - F(u)) \end{aligned} \tag{3}$$

and the second derivative:

$$\frac{d^2 z}{du^2} = O_p f(u) + U_p f(u) > 0 \tag{4}$$

The second derivative is great than zero, so z is convex. To minimize z, we make the first derivative equal to zero and the optimal value u^* satisfies:

$$F(u^*) = \frac{U_p}{O_p + U_p} = P_r(X \leq u^*) \tag{5}$$

Now, we can see that if we know the distribution of demand X and its CDF, we are able to find u^*. For example, suppose we know X follows a uniform distribution $U(a, b)$. Its CDF $F(x) = \frac{x-a}{b-a}$. Therefore $\frac{u^*-a}{b-a} = \frac{U_p}{O_p+U_p}$ and then $u^* = \frac{bU_p + aO_p}{O_p+U_p}$. For other distributions, we can follow the same way to find u^*. Even if demand X is not continuous, we can still add up the discrete probabilities to obtain u^*.

Next, we show how to find the demand distribution given user mobility data.

C. Finding Demand Distribution

In this paper, we assume that the demand of edge resources in an AoI is proportional to the number of users in that area at a certain time. The more users go to an AoI, the more edge resources need to be installed in that location.

Given the number of users in an AoI at different times, we can get data points $(x_i, y_i), i = 1, 2, \cdots$ representing the number of users and the corresponding probability. Then, we can use the fitting method [9] to find a distribution to fit these data points. For a set of data points, multiple functions can be used to fit the data. We want to find a fit that minimizes SSE and makes R^2 as close to 1 as possible. These metrics are defined as follows:

The *sum of squares due to error* (SSE) and the *total sum of squares* (SST):

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y_i})^2, SST = \sum_{i=1}^{n} (y_i - \bar{y_i})^2$$

$$R^2 = 1 - \frac{SSE}{SST} \leq 1$$

.

Here, $\hat{y_i}$ is the estimated value and $\bar{y_i}$ is the mean. The closer R^2 is to 1, the smaller the error, and the better the fitting. After we find a fitting function, we get the distribution of demand X and its CDF. Then, we can obtain the optimal number of edge resources u^*.

In the next section, we will use a concrete example to demonstrate the methods we have described here.

V. Implementation

In this section, we implement our methods in Matlab using a real trace dataset.

A. Dataset Used

We needed a dataset that could mimic user mobility, so we downloaded a trace dataset of taxi cabs in San Francisco [15]. A taxi trace was chosen because it can show user movement in a large area. The dataset contains GPS coordinates of approximately 500 taxis collected over 30 days in the Bay Area. To adapt to our implementation, we treated each taxi cab as a mobile user. Every taxi had a mobility trace file which had the following format - each line contained [latitude, longitude, occupancy, time], e.g.: [37.75134, −122.39488, 0,

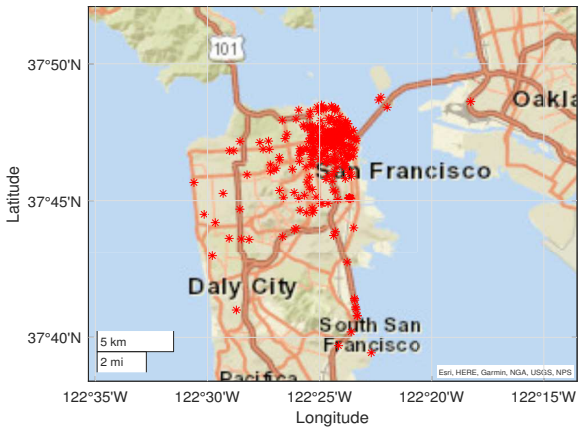

Fig. 2. Taxis in the Bay Area from the trace

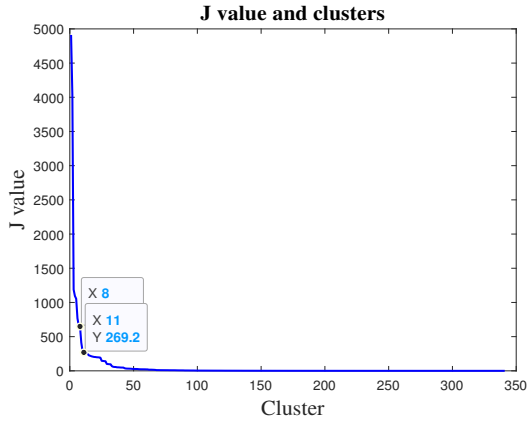

Fig. 3. Elbow in the J values

1213084687], where latitude and longitude were in decimal degrees, occupancy showed if a cab had a fare (1 = occupied, 0 = free), and time was in UNIX epoch format. We used the latitude, longitude, and time information of each taxi in our implementation; occupancy was irrelevant here.

B. Finding Areas of Interest (AoIs)

Given the taxi dataset, we wanted to identify the AoIs to install edge resources. To do that, we picked 5 observation periods everyday from May 18, 2008 to June 9, 2008 in the trace. Each observation period gave us a snapshot of the taxi locations in the Bay Area. Then, we applied Algorithm 1 to the snapshot to find the AoIs for this period. We repeated the process for all the observation periods and then settled down the final AoIs in overlapping areas.

Let us focus on one observation period [1212000000, 1212000060] ([May 28, 2008, 6 : 40 : 00 PM, May 28, 2008 6 : 41 : 00 PM]) as an example to locate AoIs. We extracted taxi locations in this period. Here, we used a 60-second time period instead of a time point because taxis may not report at exactly the time point. In that time frame, we caught 346 taxis (red star) in various locations in the Bay Area shown in Fig. 2.

Next, we fed taxi locations (data points) into Algorithm 1 to find AoIs. From the algorithm, we obtained clusters when $k = \{346, 345, \cdots, 1\}$. For each k, we calculated the J value of the clusters. After all the J values were obtained, we reverse-ordered them and then drew the relationship between k and J which is shown in Fig. 3. In the figure, the horizontal axis is the number of clusters k, starting from 1 (representing all the data points are included in one cluster) to 346 (representing each data point is a cluster in the beginning). The vertical axis is the corresponding J value. We can see that the J values of the first few ks decrease very quickly. That means it is cost-effective to continue splitting the clusters. Note that the operation is 'splitting' now instead of 'combining' because we reverse-ordered the J values. Then there is an obvious elbow. This is the cutoff section. After that, the curve becomes flat, showing the return will not be worth the cost to continue the splitting process. According to the elbow method [17], the points on the elbow, such as $8, 11$ are good heuristics for k. EIPs can decide which one to pick based on their budget, since there is usually a construction cost to deploy edge resources in a location. If the budget is high, a large number is affordable and edge resources can be installed in more AoIs. Otherwise, a smaller number can be chosen.

Finally, we settled on 11 AoIs. We represented those areas using rectangles since many streets are horizontal or vertical in the city. The format of a rectangle is [Lat_{min}, Lat_{max}, Lon_{min}, Lon_{max}]. The first element is the minimum latitude of the rectangle, the second element is the maximum latitude, the third is the minimum longitude, and the last one is the maximum longitude of the rectangle area.

C. Finding number of resources to install

In this section, we decide how many edge resources to allocate in each AoI. We take one of the AoIs we identified as an example. This AoI has a rectangle area of [37.712550, 37.764280, -122.446520, -122.391280].

To find the number of resources to install, we need to find out the distribution of demand. We do not have user demand data in the trace, so we assume the demand is proportional to the number of users and, therefore, is proportional to the number of taxis in the AoI at a certain time. To simplify, we just set the coefficient between the demand and the taxi number to one. Intuitively, the more taxis there are, the more demand there is. So now we need to find out the distribution of taxi numbers and then calculate the optimal u^* to minimize z in equation (1). If user demand data are available, the method described here still applies.

The following is the detailed process. First, we counted the number of taxis every 15 minutes from 6am to 10pm each day from May 18, 2008 to June 9, 2008 in the AoI. After we obtained all the taxi numbers, the probability of each taxi number was known. Then, the dots representing the taxi number and its probability can be drawn in Fig. 4.

Next, we tried curve fitting and obtained a normal distribution below that best fits the dots in the figure.

$$y = a * e^{-(\frac{x-b}{c})^2} \tag{6}$$

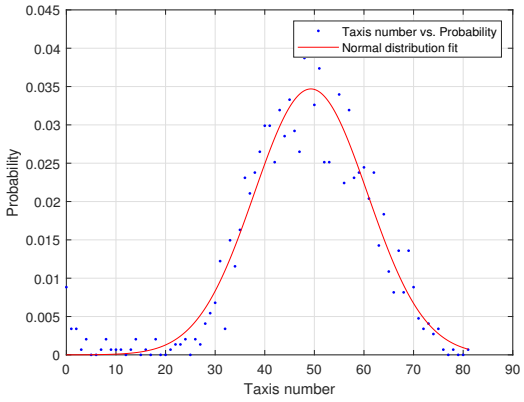

Fig. 4. Fit curve

In the above fitting function, x is the taxi number and y is the probability of the taxi number. Parameters $a = 0.0347$, $b = 49.3$, and $c = 16.15$. The fitting curve is shown in Fig. 4. The SSE of the fitting is 0.0009034 and R^2 is 0.9309, indicating that it is a good fitting.

Now that we know the demand X follows a normal distribution, its CDF is known. We are able to find the u^* in equation (5). As for U_p and O_p, the EIPs can get the values based on their business. Then $\frac{U_p}{O_p+U_p}$ is known. Let us assume $\frac{U_p}{O_p+U_p} = 0.7$. Before we use an inverse cumulative normal probability calculator to find u^*, we first need to rewrite the distribution in (6) in a standard normal distribution form as follows.

$$N(\mu, \sigma^2) = \Phi(\frac{u^* - \mu}{\sigma}) = \frac{U_p}{O_p + U_p} = 0.7$$

Here, $\mu = 49.3$, $\sigma = 11.42$. After using an inverse cumulative normal probability calculator in [2], we get $u^* = 55$. The value u^* tells us that we should install 55 edge resources to satisfy the demand of taxis (end-users) so that the expected total cost of over-deployment and under-deployment can be minimized.

In this dataset, we found a normal distribution that fits the data points well. If a continuous distribution cannot be found, the method here is still valid. We can treat probabilities as discrete values and apply summation instead of integral. If we try this in this example, the value of u^* is 54, which is very close to that obtained using the normal distribution.

Thus far, we have completed the procedure to find the number of resources in one AoI. For other AoIs, the same approach can be applied.

VI. Conclusion

In this paper, we have discussed the problems faced by Edge Infrastructure Providers (EIPs) rather than those managed by Edge Service Providers (ESPs), as is the case in most research papers on MEC. More specifically, we have gone back to the very beginning of building an MEC by asking: where to install edge resources and how many resources to install based on user movement and demand. To answer these questions, we have adopted the Hierarchical Agglomerative Clustering and Elbow methods for the first question and formulated an optimization problem to minimize the expected cost of over-deployment and under-deployment for the second question. To test the feasibility of our approaches, we have implemented the solutions using a real trace dataset of taxi cabs in San Francisco as an example. We hope the implementation can give some guidance to the EIPs to build a good edge infrastructure in the first place, so that the ESPs can further improve end-user service experience on a solid physical foundation. MEC architecture has not yet matured sufficiently. In the future, we will discuss more issues and explore solutions in MEC.

References

[1] Cluster analysis. https://en.wikipedia.org/wiki/Cluster_analysis.

[2] Inverse Normal Distribution. https://onlinestatbook.com/2/calculators/inverse_normal_dist.html.

[3] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati. Replisom: Disciplined tiny memory replication for massive IoT devices in LTE edge cloud. *IEEE Internet of Things Journal*, 3:327–338, 2016.

[4] M. T. Beck, S. Feld, A. Fichtner, C. Linnhoff-Popien, and T. Schimper. Me-volte: Network functions for energy-efficient video transcoding at the mobile edge. In *Proc. of 18th International Conference on Intelligence in Next Generation Networks (ICIN)*, pages 38–44, 2015.

[5] N. Chen, S. Zhang, J. Wu, Z. Qian, and S. Lu. Learning Scheduling Bursty Requests in Mobile Edge Computing Using DeepLoad. *Elsevier Computer Networks*, 184:33–53, 2021.

[6] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24:2795–2808, 2016.

[7] A. E.-H. G. El-Barbary, L. A. A. El-Sayed, H. H. Aly, and M. N. El-Derini. A cloudlet architecture using mobile devices. In *Proc. of IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8, 2015.

[8] Y. Jararweh et al. SDMEC: Software defined system for mobile edge computing. In *IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pages 88–93, 2016.

[9] S. Gopalakrishnan and N. Bourbakis. Curve Fitting Methods: A Survey. *The International Journal of Monitoring and Surveillance Technologies Research (IJMSTR)*, 4:33–53, 2016.

[10] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. M. Lau. Dynamic virtual machine management via approximate markov decision process. In *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2016.

[11] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues. Vehicular delaytolerant networks for smart grid data management using mobile edge computing. *IEEE Communications Magazine*, 54:60–66, 2016.

[12] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng. When mobile terminals meet the cloud: Computation offloading as the bridge. *IEEE Network*, 27:28–33, 2013.

[13] Z. Ma, S. Zhang, Z. Chen, T. Han, Z. Qian, M. Xiao, N. Chen, J. Wu, and S. Lu. Towards revenue-driven multi-user online task offloading in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 33(5):1185–1198, 2022.

[14] O. Maimon and L. Rokach. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

[15] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from https://crawdad.org/epfl/mobility/20090224/cab, February 2009. traceset: cab.

[16] S. Sardellitti, G. Scutari, and S. Barbarossa. Joint optimization of radio and computational resources for multicell mobile-edge computing. *IEEE Transactions on Signal and Information Processing over Networks*, 1:89–103, 2015.

[17] R. L. Thorndike. Who Belongs in the Family? *Psychometrika*, 18:267–276, December 1953.

[18] D. Xu and Y. Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2:165–193, 2015.