IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 26, 2020, accepted October 18, 2020, date of publication October 29, 2020,
date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034726

A Learning Based Framework for MEC Server Planning With Uncertain BSs Demands

MENGHAN SHAO[1], JIAYI LIU[2], (Member, IEEE), QINGHAI YANG[2], (Member, IEEE), AND GWENDAL SIMON[3], (Member, IEEE)

[1]State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[2]State Key Laboratory of Integrated Service Networks/Guangzhou Institute of Technology, Xidian University, Xi'an 710071, China
[3]Huawei Technologies Company Ltd., 75000 Paris, France

Corresponding author: Jiayi Liu (jyliu@xidian.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61971327.

ABSTRACT Mobile Edge Computing (MEC) architecture is composed of geographically distributed edge servers, in which computing capabilities are provisioned at the boundary of the network, which is in close proximity to the end users to provide network services with low latency. The planning of MEC edge servers at appropriate locations is the fundamental first step towards the deployment of the MEC system. In the literature, edge servers planning is based on deterministic resource requirements. This assumption largely neglects the pragmatic complexities imposed by the real dynamic world, in which base station (BS) resource demands are stochastic variables with arbitrary pattern. In view of this fact, we formulate the MEC planning problem as a joint optimization problem of MEC edge servers placement and resource allocation with uncertain BS demands through an uncertain programming formulation. Due to the complexity of this joint-uncertain problem, a learning based framework is utilized to practically solve this problem, and the relevance of applying this mechanism in practical usage with sampled arbitrary BS demands data is also discussed. Finally, we conducted intensive real-data driven simulations to evaluate the performance of our proposed mechanism. The results show the effectiveness of our approach with arbitrary BS demands.

INDEX TERMS Mobile edge computing, edge server planning, uncertain programming, learning-based framework, resource optimization.

I. INTRODUCTION

Mobile network environments often exhibit changing patterns over time. For instance, the distribution of mobile data traffic is non-homogeneous for different regions and different times of the day. Measurement studies show that the spatial inhomogeneity of traffic density in cellular network can be measured by some specific probability distributions [1]–[3]. However, these statistical parameters vary in several dimensions, such as in time and space, and they are hard to be estimated accurately. Indeed, due to the uncertainty of human behaviors, such as user mobility, the non-trivial spatial/temporal patterns also exist for the requirements of mobile infrastructure resources. The resource requirement patterns of mobile users also vary dynamically in time and space. This uncertainty makes that any infrastructure planning in mobile network is a challenging task.

The associate editor coordinating the review of this manuscript and approving it for publication was Arun Prakash.

Uncertainty has typically not been addressed in a significant planning problem in the 5G: the Mobile Edge Computing (MEC) edge server planning problem. MEC is a distributed computing architecture in which IT service environment and cloud computing capabilities are provisioned at the boundary of the network to enable enlarged processing capacity, shorter response delay, and improved user experience. Hence, it is recognized as a must paradigm for the next generation network, including the 5G [4], Internet of Things [5], and Tactile Internet [6]. The architecture of MEC is composed of geographically distributed edge servers, which are deployed at multiple locations [7]. The planning of MEC edge servers refers to the determination of the locations and resource allocation of these MEC servers, which is critical for the performance of the MEC system.

The MEC edge server planning problem is challenging. The deployment of the MEC platform depends on a number of factors, including performance criteria (such as access latency and server workload, etc.), physical deployment constraints, and various dynamic network information.

Among all these factors, the uncertain resource requirements rising from mobile end users (such as users' computing tasks) is the key driven factor for the distribution of edge servers, because the execution resources are deployed to fulfill these requirements. In the literature, a number of studies has dealt with deploying servers in the MEC infrastructure [8]–[18]. However, these existing studies assume that the resource requirements of end users are deterministic, which largely ignores the pragmatic complexity of the real dynamic world.

In recent years, deep learning based approaches have been proven to offer efficient solutions to problems that are characterized with complex uncertain inner relationships between inputs and outputs. It has been widely utilized in multiple areas in the context of intelligent 5G network: such as mobile data analytics [19], network traffic control [20], resource management [21], MEC task offloading [22] and network optimization [23]. However, to the best of our knowledge, no previous work has studied the MEC planning problem with uncertain resource demands by learning based approaches. The uncertainty comes from the facts that user demands are stochastic variables with arbitrary pattern. In this paper, we aim to address the problem of MEC edge servers planning with uncertain resource demands by the means of a learning-based approach.

We formulate the MEC planning problem through an *uncertain programming* formulation [24], in which the placement of MEC edge servers and the allocation of MEC server resources are jointly determined. In practice, mobile end users' computing demands are aggregated at the Base Station (BS) level and then dispatched towards the MEC system. Hence, BS level resource demands can represent that of the mobile users. Rather than the deterministic optimization models, in our formulation, the BS demands are treated as uncertain random variables with arbitrary pattern, and the objective is to minimize the expectation of access latency. Due to the complexity of solving such a joint-uncertain problem, we design a learning-based framework, which integrates stochastic simulation, neural network, and genetic algorithm (GA). We utilize this framework to practically solve this problem by accelerating the searching in the solution space. We also discuss the relevance of applying this mechanism in practical usages with sampled arbitrary BS demands data to highlight the pragmatism of our approach. To evaluate the performance of the learning-based framework on planning MEC servers with realistic BS demands, we conducted intensive simulations: a set of small network scale simulation with lognormal BS demands, and a set of real-data driven large network scale simulation with historical BS demands. The results show the effectiveness of our solution on planning edge servers with arbitrary BS demands by comparing to several baseline algorithms.

The remainder of this paper is structured as follows. The related research works are discussed in Section II. The joint-uncertain MEC server placement problem is formulated in Section III. Then, we introduce the learning based framework to practically solve the problem in Section IV. The simulation results are discussed in Section V. Finally, the paper concludes with some insights and future research work in Section VI.

II. RELATED WORK

By bringing computing capacities and network functions to the network edge, MEC is expected to provide low latency services for mobile users. This attractive architecture has thus received a considerable attention in the recent years. Researchers have identified the fundamental problem of MEC planning, which is to determine the best locations for the MEC resources with respect to the requirements of end users. We review this literature in the following.

A. VIRTUAL RESOURCE PLACEMENT

A number of works have studied the provisioning and placement of resources for virtual network functions (VNFs), services and virtual machines, and so on. The authors in [25] study the placement of edge VNFs in an edge infrastructure by taking service end-to-end latency and network dynamics into consideration. An optimal application placement mechanism on an energy-limited edge servers to maximize the QoS of the system is designed in [26]. Similarly, the authors in [27] propose a novel dynamic user-managed service placement mechanism to overcome the unavailability of future information and unknown system changes. In [28], the authors address the collaborative service placement in MEC environment and propose an efficient decentralized algorithm while taking service heterogeneity, spatial demand coupling and decentralized coordination into account.

The optimal placement of virtual machines among MEC servers has also been widely studied. An optimal solution in allocating the virtual resources is investigated to cope with the non-uniform distribution of signaling messages and the irregularity of network topologies by means of Schwartz-Christoffel conformal mappings in [29]. The authors in [30] find the optimal placement of virtual machine replica copies to minimize the average response time in an MEC architecture, supporting various requests demand among multiple applications with limited capacity of servers in edge networks. The works in [31] and [32] mainly focus on reducing resource consumptions of virtual machines. An appropriate job layout method is presented in [31] to allocate virtual machine requests to minimize the energy consumption of MEC servers. The authors in [32] study both the virtual machine placement and workload assignment in MEC. Their work shows the influence of latency requirement of applications, capacity of MEC servers and request workload of users on hardware consumptions.

These works assume that the physical MEC infrastructure has been deployed and investigate the virtual resource allocation and service configuration problem. However, the placement of virtual machines is based on the deployment of physical infrastructure, which is the fundamental first step to establish an MEC platform. Therefore, the deployment

TABLE 1. Comparison of edge server placement algorithms.

Paper	Shared workload	Clustering	Co-located	Number of servers	Capacity	Latency
B. Li et al.[8]	no	yes	yes	fixed	limited	minimized
S. Wang et al.[9]	no	yes	yes	fixed	limited	minimized
Y. Li et al.[10]	no	no	yes	fixed	limited	threshold
H. Yin et al.[33]	no	no	no	unfixed	limited	minimized
T. Lahderanta et al.[11]	share	yes	yes	fixed	limited	minimized
J. Meng et al.[34]	share	no	yes	fixed	limited	minimized
Y. Li et al.[12]	no	no	no	fixed	no	threshold
K. Xiao et al.[13]	no	no	yes	fixed	limited	minimized
G. Manasvi et al.[14]	share	no	yes	fixed	limited	minimized
S. Lee et al.[17]	no	yes	yes	minimized	limited	threshold
T. Chin et al.[15]	no	yes	yes	fixed	no	not used
X. Xu et al.[18]	no	yes	yes	fixed	limited	minimized

of physical edge servers has attracted more research interest recently.

B. PHYSICAL RESOURCES PLACEMENT

We summarize related works on the topic of physical edge server placement in Table 1. The algorithms are described with the following syntax. Shared workload denotes whether the workload is sharing between servers in the algorithm. Similarly, Clustering denotes whether the algorithm is clustering-based or not. And we also compare the roles of the number of servers, capacity and latency of different algorithms. Most of the works consider the edge server placement as a placement problem by selecting K locations from N candidates with different objections. In this case, generally the edge servers are co-located with access points [8]–[16] or other network functions [17], [18]. These works are marked as *yes* in the column of Co-located.

Some of these works focus on reducing the average transmission latency or physical distance between servers and BSs. The authors in [8] analyze workload and task requirements of BSs in mobile edge computing environments and propose a K-means-based algorithm for selecting appropriate cells to deploy edge servers. In [9], the authors formulate the problem as a mixed integer programming problem, aiming to find out strategic locations for edge servers to minimize the access delay and balance the workload of edge servers. The authors in [11] develop an edge server placement scheme, which aims to minimize the distance between access points and edge servers. They formulate this problem as a capacitated location-allocation problem and evaluate it in edge computing and fog computing environments respectively. Based on the available information gathered from the overlay social network groups, an edge sever placement strategy is proposed in [13] to identify a number of critical BSs to place edge servers while supporting low latency communication. To ensure the latency between a server and a BS, the authors in [14] conduct a study of deploying edge servers in heterogeneous edge computing systems to minimize the expected response time while considering the response time fairness of each single BS.

Another research direction that has received attention is related to the reduction of the number of servers and energy consumption. The authors in [17] model the MEC server placement as a capacitated clustering problem and propose an heuristic algorithm that assures a certain delay with the minimum number of placed MEC servers. The authors in [10] and [18] both formulate the problem as a multi-objective optimization problem, [10] tries to find out an energy-aware edge server placement method to reduce the total energy consumption while keeping an acceptable access delay. In [18], the authors study the actual traffic flow of a city to search for the specific quantity and locations of edge servers with low latency, balanced workload and minimum number of edge servers. The authors in [12] address the challenge of server placement based on resource requirements forecasting and propose a cross-regional resources optimization algorithm with the goal of minimizing the cost of service providers.

In addition, server placement and resource allocation are simultaneously considered in some works. In [15], the edge server placement and work allocation strategy is investigated with the goal of minimizing traffic load in the mobile edge networks for green communications. For joint edge server placement and application configuration, a local-search based algorithm to minimize the service cost and opening cost of edge servers is proposed in [34]. In [16], the authors study the joint computational services and physical resources allocation problem for latency sensitive applications in MEC. They combine the computation partitioning and two-dimensional resource allocations in both computation resource and network bandwidth.

We also identify some studies related to workload sharing. The works in [11], [14], [34] consider sharing the workload between edge servers, which means that the demands of an access points can be offloaded towards more than one edge servers. It is beneficial to share computing tasks to balance the workloads of edge servers. The authors in [33] present a framework, called Tentacle, to discover unforeseen locations for edge servers to explore opportunities and find new possible server locations to ensure the proximity between users and edge servers. Following these works, we also consider the workload sharing among edge servers and unconstrained server locations.

Our present work differentiate from these previous papers by taking into account the uncertain spatio-temporal distribution of requests in different BSs, which is an

absolutely key factor for the deployment of MEC servers. Taking this pragmatic issue into consideration, we propose a learning-based framework to discover optimal locations for edge servers and allocation server workloads with uncertain BSs demands. To the best of our knowledge, the current work is the first to apply uncertain programming in MEC environments. The proposed solution copes with the deployment of MEC servers with uncertain BS demands, an issue which has been overlooked in previous relevant research works.

III. PROBLEM FORMULATION

We consider an MEC system in which geographically distributed edge servers form an intermediate computation layer between the BSs (end devices) and the remote center cloud for the purpose of reducing latency, ensuring efficient network services, and offering improved user experience. The system architecture is depicted as in Figure 1. User computation demands are first gathered at the BS level, and then dispatched and served by MEC edge servers. Hence, we distinguish two representative layers: the BS layer includes all mobile BSs with arbitrary computation resource demands, and the MEC layer contains the MEC edge servers that provide the computing services towards the BSs.

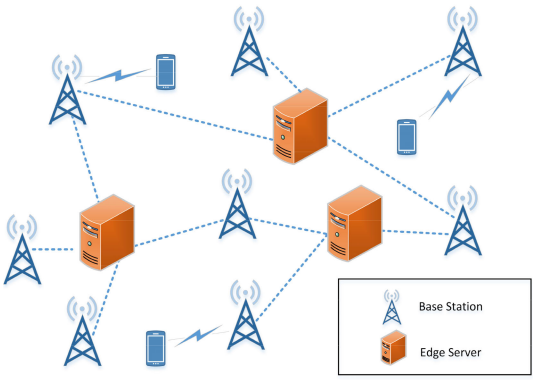

FIGURE 1. Mobile edge computing architecture.

The selection of MEC infrastructure sites' locations is the first step of building an MEC system. The planning of MEC infrastructure sites determines the physical locations of edge servers, as well as the allocation of server computing resources to provide services towards BSs. The problem is essentially driven by the stochastic BS resource demands, which brings difficulties in practically solving the problem with low complexity. Different from existing works with *a-priori* knowledge on BSs demands, we formulate a joint-uncertain problem for determining the placement of edge servers and the allocation of server resources under uncertain BS demands.

A. ASSUMPTIONS

Before modelling the edge server planning problem, we first make some reasonable assumptions as follows.

1) We do not restrict the location of the edge servers. Some works model the placement problem as selecting some locations from a set of candidate ones. As claimed in [33], it prevents the exploration of better locations for edge servers. Hence, we consider that edge servers can be freely deployed in any position in a given region.
2) Sharing workload between edge servers is known to improve the trade-off between service proximity and server workload balance. We adopt this model in which the resource demands of BSs, which are originated from end users, can be dispatched towards multiple edge servers for workload sharing.
3) Due to the variation of mobile user behaviors, the demands of BSs change dynamically in space and in time. Thus we model the service requests of each BS as an arbitrary stochastic variable.

B. BS-SERVER MODEL

Within a given range R, an MEC platform composed of N edge servers provides edge computing services towards M BSs. The BSs receive end users' computing requests and offload their requests towards the edge servers.

- The set S denotes the set of mobile edge computing servers, with $s_i \in S$ denotes one edge server, $1 \leq i \leq N$. Each edge server has limited computing resources. We use c_i to denote its maximum capacity to process computing tasks.
- The set B denotes the set of BSs, with $b_j \in B$ denotes one BS, $1 \leq j \leq M$. We assume that the servers and BSs are distributed within R which is represented by a two-dimensional plane, the coordinate point $(p_j, q_j) \in R$ indicates the position of the BS b_j on the plane.
- Correspondingly, BS b_j issues a resource demand v_j, which in practice is originated from end users connected through wireless links. The demand v_j is assumed to be an uncertain stochastic variable. The BS demands vector $\mathbf{V} = \{v_1 \cdots v_j \cdots v_M\}$ is composed of all BSs demands.

TABLE 2. Notations.

Symbol	Meaning
S	Set of MEC servers to be placed in the network
B	Set of BSs in the network
N	Number of MEC servers
M	Number of BSs
(p_j, q_j)	Location of BS b_j
v_j	Resource demand of BS b_j
d_{ij}	Spatial distance between BS b_j and edge server s_i
(x_i, y_i)	Decision variable that indicates the location of edge server s_i
z_{ij}	Decision variable that represents resources edge server s_i supplied to BS b_j

C. PROBLEM FORMULATION

1) *Variables*: Two sets of decision variables are defined to jointly formulate the edge server placement and server resource allocation problem.

- Firstly, a couple of real variables $(x_i, y_i) \in R$ denote the 2D coordinate points on which the edge server s_i is placed.
- Secondly, a real variable z_{ij} represents the computing resources supplied by edge server s_i to BS b_j.

2) *Objective Function*: Our objective is to find the optimal locations to deploy a fixed number of edge servers to minimize the distance to their associated BSs, while satisfying the capacity constraints. For each requests variable vector $\mathbf{V}$, the total distance of the whole system weighted by the allocated servers workloads can be expressed as

$$C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V}) = \sum_{i}^{N} \sum_{j}^{M} z_{ij} d_{ij}. \tag{1}$$

Here, we use the Euclidean distance to represent the spatio-distance between the BS b_j and the edge server s_i, then we obtain the expression $d_{ij} = \sqrt{(x_i - p_j)^2 + (y_i - q_j)^2}$. Therefore, the weighted total distance is

$$C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V}) = \sum_{i}^{N} \sum_{j}^{M} z_{ij} \sqrt{(x_i - p_j)^2 + (y_i - q_j)^2}. \tag{2}$$

The total conveying distance $C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V})$ is a stochastic variable due to the fact that requests are stochastic variables. Our objective is to find a location vector $(\boldsymbol{x}, \boldsymbol{y})$, and the resource allocation $\boldsymbol{z}$ to minimize the expectation of $C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V})$:

$$\min_{\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}} E[C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V})] \tag{3}$$

3) *Constraints*: In order to obtain an feasible placement $(\boldsymbol{x}, \boldsymbol{y})$ and resource allocation $\boldsymbol{z}$, we consider the following constraints.

- For all BSs and edge servers, the quantity of computing resources supplied by edge server s_i to BS b_j must be positive:

$$z_{ij} \geq 0, \quad i \in [1, N], j \in [1, M]. \tag{4}$$

- For each BS b_j, its resource demand must be satisfied by all edge servers:

$$\sum_{i=1}^{N} z_{ij} = v_j, \quad j \in [1, M]. \tag{5}$$

- For each edge server s_i, the amount of resources provided by a server for all base stations must be less than its own capacity:

$$\sum_{j=1}^{M} z_{ij} \leq c_i, \quad i \in [1, N]. \tag{6}$$

- Each edge server s_i should be placed within the given range R:

$$(x_i, y_i) \in R, \quad i \in [1, N]. \tag{7}$$

Above all, we formulate the edge server placement problem as follows:

$$\min_{\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}} E[C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V})] \tag{8}$$

Subject to:

$$\begin{cases} z_{ij} \geq 0, & i \in [1, N], j \in [1, M]. \\ \sum_{i=1}^{n} z_{ij} = v_j, & j \in [1, M]. \\ \sum_{j=1}^{m} z_{ij} \leq c_i, & i \in [1, N]. \\ (x_i, y_i) \in R, & i \in [1, N]. \end{cases} \tag{9}$$

In essence, the problem is to obtain the coordinates of the edge servers in the potential region and allocate servers resources towards BSs by considering uncertain BS demands, which can be solved by the theory of uncertain programming.

IV. THE LEARNING-BASED SERVER PLACEMENT FRAMEWORK

The complexity of the above problem is high. The placement of the edge servers can be reduced to the classical capacitated facility location problem, but it is NP-Hard. Hence it is difficult to solve the problem with traditional methods as it is a complex joint optimization problem with uncertain variables as input data. The uncertainty of variables and the joint optimization problem should be considered simultaneously, which greatly increases the difficulty of solving the problem. In the following, we introduce a learning-based framework to solve the problem, and discuss its practicability.

A. THE OVERALL SOLVING PROCESS

We propose a learning-based framework integrating stochastic simulation, neural network, and GA to solve the joint-uncertain programming problem, which has a huge search space. The flow chart of the framework is shown in Figure 2. We first decompose the original joint problem into two subproblems: the edge server location subproblem and the server resource allocation subproblem. For the server location problem, we utilize the GA to search the solution space and obtain the optimal locations of edge servers as GA can rapidly search for the approximate global optimum under complicated design environment. The location of servers, i.e. the value of coordinates of $(\boldsymbol{x}, \boldsymbol{y})$, act as chromosomes. For a given $(\boldsymbol{x}, \boldsymbol{y})$, we estimate the value of the uncertain function $E[C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V})]$ to obtain the fitness function of each chromosome in GA to select the best chromosome.

Owing to the uncertainty of BS demands, we can get an approximate value of $E[C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}|\boldsymbol{V})]$ by sampling from the uncertain demands space numerous times and taking the average value. Stochastic simulation has a good performance and wide application when faced with significant uncertain problem, thus we adopt stochastic simulation to calculate

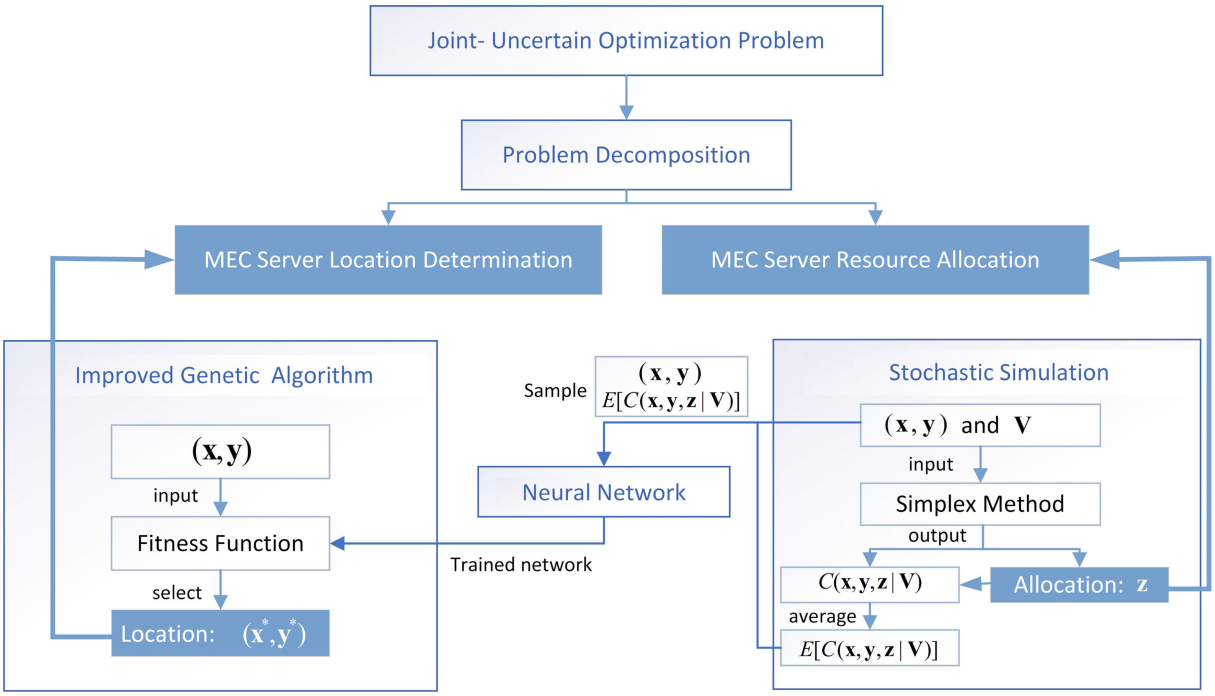

FIGURE 2. The learning-based server placement framework.

$E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$. Within the process of stochastic simulation, with each sampled BS demands and given server locations, we solve the resource allocation subproblem, which is essentially a linear programming problem, by the simplex algorithm. Once the resource allocation vector z is obtained, we can easily get the value $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$ and finally $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ can be gained by calculating average value of $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$.

The huge search space of the solutions and the complexity of the stochastic simulation deteriorate the performance of the overall solving process. We thus introduce a neural network, which is embedded into the GA, to accelerate the searching process. We utilize the samples of the stochastic simulation and corresponding calculated fitness function to train the neural network. Then, with the neural network, the value of $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ can be directly obtained, given the corresponding objective value of any edge server locations. Overall, the framework consists of three main bodies:

- Monte Carlo stochastic simulation to get a set of samples $(\boldsymbol{x}, \boldsymbol{y})$ and the corresponding $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ for the purpose of training the neural network. During each iteration of the simulation, we use simplex method to solve the linear programming resource allocation subproblem and get the solutions of z. Then we calculate the corresponding $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ for a given location vector $(\boldsymbol{x}, \boldsymbol{y})$.
- The neural network, which is trained by the samples from stochastic simulation. The location vector $(\boldsymbol{x}, \boldsymbol{y})$ and the value of $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ are regarded as inputs and outputs respectively. The neural network accelerates the exploration of the search space of the edge server location subproblem.
- GA to obtain the optimal locations for servers, upon the training of the neural network. The GA is initialized by a random population, then during each iteration, we use the neural network to compute the fitness function to achieve the aim of accelerating the search process.
- After the optimum edge server locations is determined, we can determine the corresponding resource allocation.

Then, we introduce each module of the learning-based framework in details.

B. STOCHASTIC SIMULATION

Monte Carlo stochastic simulation is regarded as a good solution when faced with significant uncertainty in the process of making a forecast or estimation, rather than just replacing the uncertain variable with a single average number. It is a technique for depicting sampling experience, which mainly relies on the statistics of probability distribution and the modeling of sampling variables. By taking enough samples from the probability distribution, an uncertainty problem can be transformed into multiple certainty problems.

In our model, the demand of each BS is an uncertain variable, which results in the workload-weighted distance $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$ being an uncertain variable. Hence we use stochastic simulation to obtain the expectation value of $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$. Given an arbitrary position coordinate vector of servers $(\boldsymbol{x}, \boldsymbol{y})$, we sample the variables $\boldsymbol{V}$ from its distribution

multiple times to calculate $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$. It means we must handle the following functions:

$$U : (\boldsymbol{x}, \boldsymbol{y}) \rightarrow E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})] \tag{10}$$

Every time a sample is taken, the corresponding value of $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$ should to be calculated. Note that this model is different from traditional stochastic programming models because there is a resource allocation subproblem in it:

$$\begin{cases} \min\limits_{\{z\}} \sum\limits_{i}^{N} \sum\limits_{j}^{M} z_{ij} \sqrt{\left(x_i - p_j\right)^2 + \left(y_i - q_j\right)^2} \\ z_{ij} \geq 0, \quad i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, \mathrm{m} \\ \sum\limits_{i=1}^{n} z_{ij} = v_j, \quad j = 1, 2, \ldots, m \\ \sum\limits_{j=1}^{m} z_{ij} \leq c_i, \quad i = 1, 2, \ldots, \mathrm{n} \end{cases} \tag{11}$$

This subproblem is essentially a transportation problem and belongs to linear programming, thus we can employ simplex algorithms to solve this sub-problem.

The stochastic simulation process is described in detail in Algorithm 1. Stochastic simulation consumes a large amount of computation time, because in each iteration the uncertain function is calculated with sufficient enough samples, and for each sample the resource allocation subproblem needs to be solved. To overcome this drawback, the neural network is utilized to reduce the computational complexity and speed up the solution searching process.

Algorithm 1 Stochastic Simulation

Input:

The potential feasible region R;

The probability distribution of $\boldsymbol{V}$;

Output:

Server location $(\boldsymbol{x}, \boldsymbol{y})$ and its corresponding objective function $E(C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V}))$.

1: Generate location coordinates (x_i, y_i) randomly for each server according to a certain rule from the potential feasible region of the inspection.
2: Generate random variable v_j from the probability distribution of demands for each base station.
3: Solve linear programming by simplex algorithm and obtain its optimal target value $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$.
4: Update $U \leftarrow U + C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$.
5: Repeat steps 2-4 for N times.
6: Calculate the average value $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$, i.e.$U/N$ to obtain $E(C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V}))$.;

C. NEURAL NETWORK TRAINING

We use artificial neural network to approximate the value of $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$. For the training, we define coordinate vector of the servers location $(\boldsymbol{x}, \boldsymbol{y})$ as the input and $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ as the output of the network. For instance, $(\boldsymbol{x}, \boldsymbol{y})$ and its corresponding function value $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ is considered as a sample of neural network. After the network is trained, once $(\boldsymbol{x}, \boldsymbol{y})$ is given, the expected value of $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$ can be obtained rapidly without stochastic simulation. By embedding the trained neural network into the genetic algorithm, the fitness function can be calculated with greatly improved efficiency, thus the performance of the whole solving process is improved. The overall steps for training a neural network are as follows:

1) Generate input and output data for uncertain functions through stochastic simulation described in Section IV-B, $(\boldsymbol{x}, \boldsymbol{y})$ are input neurons and the value of $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ are output neurons.
2) Normalize the input and output data with a linear function and use the normalized data as training samples.
3) Use the above training samples to train a back-propagation (BP) neural network to approximate the uncertain function $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$.

D. IMPROVED GENETIC ALGORITHM

We adopt the application of GA in the learning-based framework as GA can rapidly search for the approximate global optimum under complicated design environment. Comparing to the general GA, our proposed algorithm improves the computation efficiency with the neural network as described previously, which makes it possible to deal with large-scale optimization problems efficiently. The procedure are described as follows. Firstly, we generate server location vectors $(\boldsymbol{x}, \boldsymbol{y})$ randomly as the initial population. Then different from the general method, the neural network is used to calculate the objective values $E[C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})]$ to accelerate the computation of the fitness function. After selection, crossover and mutation on the current population, and the optimal result is obtained after the algorithm converges. The specific description of the improved GA is shown in Algorithm 2.

E. PRACTICAL RELEVANCE

This learning-based framework is general in solving a series of practical problems with uncertain input parameters. In practice, historical BS demands data can be utilized in the place of the samples generated in the stochastic simulation part to estimate the effective expectation of demands-weighted distance $C(\boldsymbol{x}, \boldsymbol{y}, z|\boldsymbol{V})$, which essentially solve problems with real settings and arbitrary demands distributions. In Section V, we conducted two sets of simulations: a small scale simulation with samples generated through stochastic simulation method; and a large scale real-trace-driven simulation in which real BS demands data are utilized as samples.

Moreover, multiple modules in the framework can be replaced by mechanisms with similar functionalities. For instance, (1) the GA in the framework can be replaced by other improved meta-heuristic algorithms to further improve the problem solving efficiency; (2) stochastic simulation can be substituted by other methods to estimate the value of the uncertain function; (3) the simplex algorithm can be replaced by other optimization algorithms as long as it can be used to solve the corresponding subproblems; and (4) the BP network

Algorithm 2 Improved Genetic Algorithm Based on Neural Network

Input:

The potential feasible region R;

The trained neural network in Section IV-C.

Output:

The optimal position coordinates of servers $(\boldsymbol{x}^*, \boldsymbol{y}^*)$.

1: Generate randomly coordinates (x_i, y_i) from the potential feasible region $\boldsymbol{R}$ of the inspection, initialize chromosomes $\boldsymbol{Q}^k = (\boldsymbol{x}^k, \boldsymbol{y}^k) = \{x_1^k, x_2^k, \ldots, x_N^k, y_1^k, y_2^k, \ldots, y_N^k\}$, $k = 1, 2, \ldots, pop\_size$, which denote locations of all the servers.

2: Calculate the objective values for all chromosomes, $\boldsymbol{Q}^k = (\boldsymbol{x}^k, \boldsymbol{y}^k) = \{x_1^k, x_2^k, \ldots, x_N^k, y_1^k, y_2^k, \ldots, y_N^k\}$, $k = 1, 2, \ldots, pop\_size$. Here,we define the objective value of chromosome $(\boldsymbol{x}, \boldsymbol{y})$ as $E[C(\boldsymbol{x}, \boldsymbol{y}, z|V)]$, which is calculated by the neural network already trained in Section IV-C instead of stochastic simulation.

3: Compute the fitness function for each chromosome $\boldsymbol{Q}^k$, $k = 1, 2, \ldots, pop\_size$, the rank-based evaluation function is defined as $\text{Eval}(Q^k) = \beta(1-\beta)^{k-1}$, $k = 1, 2, \ldots, pop\_size$, where the chromosomes $\boldsymbol{Q}^1, \boldsymbol{Q}^2, \ldots, \boldsymbol{Q}^{pop\_size}$ are assumed to have been rearranged from good to bad according to their objective values and $\beta \in (0, 1)$ is a parameter in the genetic system.

4: Select new population from the initial population. The selection process is based on spinning the roulette wheel which is characterized by the fitness of all chromosomes for pop_size times, and each time we choose a single chromosome. Thus we update the pop_size chromosomes $\boldsymbol{Q}^k, k = 1, 2, \ldots, pop\_size$.

5: Update the chromosomes $\boldsymbol{Q}^k, k = 1, 2, \ldots, pop\_size$ by inversion operation. The parameter P_c is noted as the probability of crossover of a genetic system. This probability gives us the expected number $P_c \cdot pop\_size$ chromosomes undergoing the inversion operation.

6: Update the chromosomes $\boldsymbol{Q}^k, k = 1, 2, \ldots, pop\_size$ by mutation operation. The parameter P_m is the probability of mutation, which gives us the expected number of $P_m \cdot pop\_size$ chromosomes undergoing the mutation operations.

7: Repeat 2 to 6 steps for a given number of cycles.

8: Return the best chromosome $\boldsymbol{Q}^* = (\boldsymbol{x}^*, \boldsymbol{y}^*)$ as the optimal locations for servers;

can be further replaced by more sophisticated deep learning models to improve the prediction accuracy. Finally, with generality and universality, this framework can be used to solve a series of uncertain problems with practical considerations.

Besides, we analyze the runtime complexity of our framework and K-means.GA is essentially the main body of the framework to search optimal locations for MEC servers, while stochastic simulation is used to produce samples for training neural network and neural network is used to accelerate the process of GA. Hence, the complexity of the solution mainly depends on GA. The runtime complexity of GA is decided by the maximum permission iterative number of times and genetic operator including selection, crossing and mutation. We assume that the population size of GA is POP_SIZE and the number of MEC servers is N, then the runtime complexity of crossing and mutation operation is $O(N * POP\_SIZE)$ and $O(POP\_SIZE)$ respectively, and the runtime complexity of selection operation is less than $O(POP\_SIZE^2)$. Thus the complexity of GA is $O(MAX\_GEN * (N * POP\_SIZE + POP\_SIZE + P\,OP\_SIZE^2))$, which finally can be reduced to $O(N * POP\_SIZE)$ since the MAX_GEN(maximum generations) is a constant and POP_SIZE is tunable. As for K-means, the runtime complexity is $O(M \log M)$, where M means the number of BSs.

V. EVALUATION

We implemented the proposed learning-based framework and designed experiments to evaluate the performance of the framework on planning MEC servers. We first show the performance of our framework in a small scale network with generated simulation data, then we test the effectiveness of our framework based on a real data set from Shanghai Telecom [35]. We conduct the simulation on a PC equipped with Intel Core i3, 4 core processor, and 12 GB RAM and the software simulation environment of the proposed framework is based on ECLIPSE, MATLAB and IBM CPLEX. Stochastic simulation is implemented by java language, and IBM CPLEX is used in the third step of stochastic simulation to realize the simplex algorithm. GA and neural network is implemented with the MATLAB mathematical toolbox as it can reduce programming difficulty. The simulation environment settings are described as follows.

A. SIMULATION PARAMETERS SETTING

1) STOCHASTIC SIMULATION SETTING

Monte Carlo stochastic simulation calculates the value of uncertain function by sampling from uncertain variables a large number of times. The stable value can be obtained by averaging the estimated values of statistics by increasing the number of samples. The sampling times should be large enough to ensure the performance of stochastic simulation. However, over-sampling greatly increases computation complexity, hence it is important to select an appropriate sampling times. In order to find the appropriate sampling times, we measured the relationship between sampling times and relative error. The relative error, with respect to sampling 4000 times, is calculated as:

$$\text{relative error} = \left| \frac{E[C(\boldsymbol{x}, \boldsymbol{y}, z|V)]_N - E[C(\boldsymbol{x}, \boldsymbol{y}, z|V)]_{4000}}{E[C(\boldsymbol{x}, \boldsymbol{y}, z|V)]_{4000}} \right|$$

where $E[C(\boldsymbol{x}, \boldsymbol{y}, z|V)]_N$ represents the uncertain function value obtained by sampling N times. We selected 4000 as a baseline sampling times because it provides a

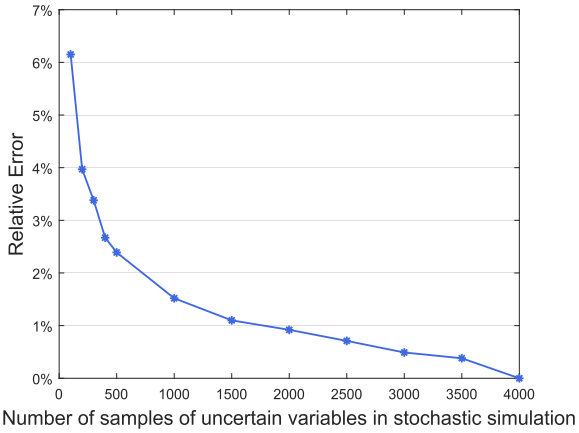

FIGURE 3. Variation of the relative error with sampling times in Monte Carlo stochastic simulation.

qualified estimation. As shown in Figure 3, when the sampling times reach 2000, the relative error is less than 1 %, thus we chose 2000 as the number of samples.

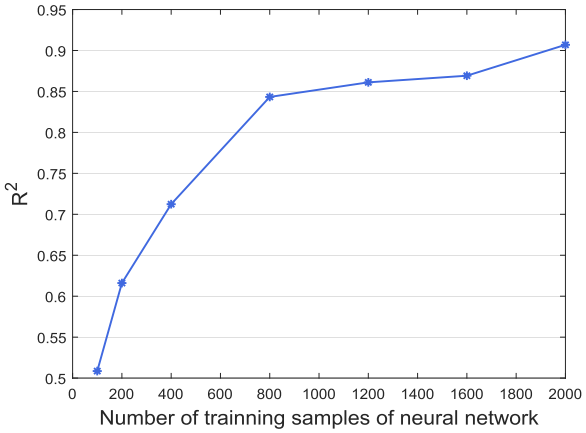

FIGURE 4. Change of R^2(coefficient of determination) of neural network with the number of training samples.

2) NEURAL NETWORK SETTING

As mentioned in Section IV-C, we adopt BP neural network to fit the uncertain function to accelerate the genetic algorithm. In order to ensure the prediction accuracy of neural network, we must use enough samples to train the neural network. The coefficient of determination R^2 is used to measure the fitting accuracy of neural network, and we investigated the variation of R^2 with the number of training samples. As shown in Figure 4, R^2 is greater than 0.9 when the sample reaches 2000, which we interpreted as a good enough prediction accuracy of neural network. Therefore, we took 2000 as the number of input samples of the neural network.

3) GENETIC ALGORITHM SETTING

The improved GA is deployed to search the solution space to obtain the best locations for MEC servers. As described earlier in Section IV-A, MATLAB GAOT toolbox is used to implement GA. Normalized geometric selection was chosen to select as it has a shorter compilation time. We choose arithmetic crossover as the crossover procedure because point crossover is too simplistic to work effectively and arithmetic crossover procedure is usually the ideal crossover option for use in projects. Non-uniformly distributed mutation operator was chosen as the mutation operator because it is considered to function well with multiple variables. To present the solution convergence, we show the change of fitness function with the number of generations in Figure 5, and the BS demands and MEC servers setting are the same in large scale simulation that will be introduced later. It can be seen from the picture that the curve of fitness function become smooth after 80 generations, thus we set the number of generation as 100.

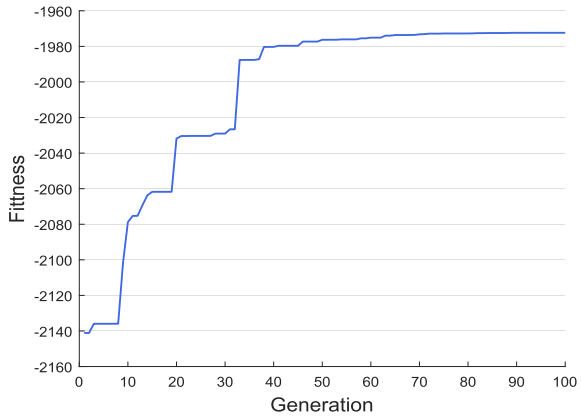

FIGURE 5. The fitness function of GA with the number of generations.

B. COMPARED ALGORITHMS

To evaluate the performance of our proposed framework, we compared our framework with other edge server placement methods:

1) *Random Fit Algorithm*: This algorithm randomly selects a coordinate point on the plane as the placement location of each edge server.
2) *Top-first Algorithm*: Due to the inconformity of BSs demands, we choose the BSs with heaviest demands as the deployment locations of the edge servers.
3) *K-means Algorithm*: The purpose of K-means clustering algorithm is to minimize the sum of the distances between all data points and their associated cluster centers. We regard BSs as the data points and the center of the cluster is selected as the location of the server.

C. SMALL SCALE SIMULATION

We first conducted a small scale simulation based on generated simulation data. As the first step of simulation, some basic parameters such as location and demand of each BS should be given for selecting locations for MEC servers, thus we adopt the following BS location and BS demand models:

1) **BS location**: The inspection region is a $L \times L$ square on which all BSs scattered. We adopt homogeneous spatial poisson point process (SPPP) (a widely applied models for modeling BS locations [36], [37]) to generate coordinates of BSs in the given region.
2) **BS demands**: The statistically spatial and temporal BS demand distribution is non-uniform across different regions in different time, making the resource demands of each BS an uncertain variable. Our framework can be applied to arbitrary BS demands. To conduct an illustrative simulation with reasonable setting, it is vital to model the demands of BSs. Firstly, we assume that the traffic load of each BS represents BS demand, then we model BS demand by following the model represented in [1].
Generally, BS traffic distribution shows a strong non-uniform characteristic due to the changes of user behaviors, which impacts both temporal and spatial traffic distributions of BS. In time domain, the traffic volume of a region changes periodically with day-night behaviors of users, which leads to periods of low traffic and high traffic. Based on the actual data of the current cellular network, a *sinusoidal superposition* model is utilized to approximate the time traffic changes of multiple BSs in a given region. In spatial domain, because users' social behaviors are similar in the same area, traffic distributions has a strong correlation with region types. The *lognormal* distribution is adopted to depict the spatial inhomogeneity of traffic in cellular networks. Taking into both temporal and spatial domain in consideration, a combined spatial-temporal traffic modeling method is utilized for describing the traffic pattern of a single BS, which both reflects the periodicity and randomness. The traffic value of each BS in the region at each time is generated by lognormal distribution with parameters $\mu(t)$ and σ.

$$V_j(t) = \log nrnd(\mu(t), \sigma)$$

In this expression, $V_j(t)$ is the demand of BS b_j at time t: $\mu(t)$ is a parameter related to the average traffic volume of all BSs in the region and presents periodic variation with time; σ is different correlated with the region type. We follow the parameters setting in [1]. As depicted in Figure 6, the change periods of $\mu(t)$ is 24 hours, which is consistent with human daily activities. The value of σ varies with different regions, here we set the value of σ to 1.3. The probability density function of resource demands at spare time ($\mu = 4.19$) and busy time ($\mu = 5.56$) is shown in Figure 7. Finally, Figure 8 demonstrates the distribution of BSs and their corresponding resource requirements at a specific time $t = 20$. In practical usages, we can also use the historical BSs demands data, instead of this spatio-temporal model.

In order to assess the learning-based MEC server placement framework, the average distance between BSs and the

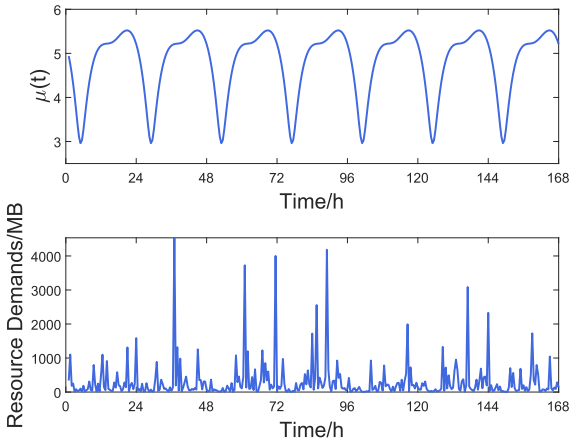

FIGURE 6. Demands variation of a single BS with time series.

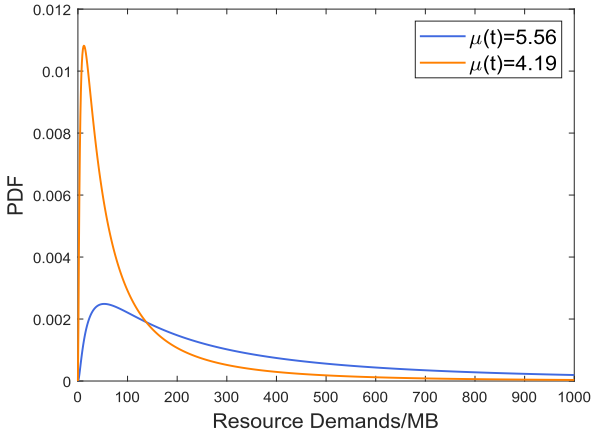

FIGURE 7. Probability density distribution of demands at different time.

server weighted by BSs demands is evaluated as in [11], which is defined as:

$$\text{distance} = \frac{\min \sum_i^N \sum_j^M z_{ij} \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2}}{\sum_{j=1}^M v_j}$$

We set three edge servers to serve BSs with layout as shown in Figure 8. We show the location of servers placed by our proposed solution and other algorithms in Figure 9. Then, we investigate the variation of demands-weighted distances of different solutions with the number of BSs. In order to avoid the contingency in the result, we generated ten groups of BS locations and take the average value. Table 3 and Figure 10 illustrates the performance of the learning-based framework (LBF) and other algorithms. With the number of BSs increasing, the weighted distance, which also represents latency, of Top-first, K-means and LBF goes up, and that of the Random algorithm shows a trend of fluctuation. The demands-weighted distance of our learning-based framework is on average 19.43 % less than K-means, 33.61 % less than Top-first, and 44.45 % than Random. Therefore, our solution has a better performance than the other approaches in

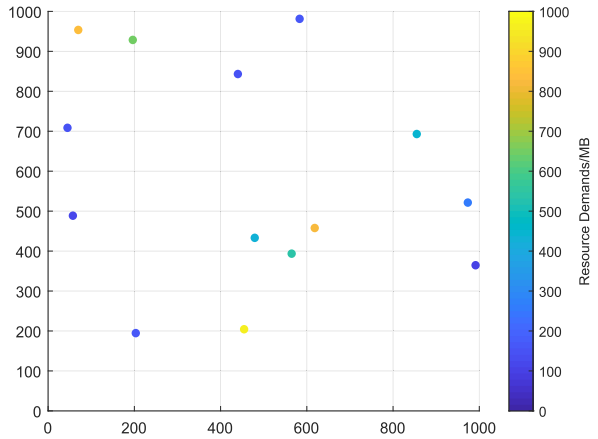

FIGURE 8. Base stations distribution and corresponding demands at time t=20 in small scale simulation.

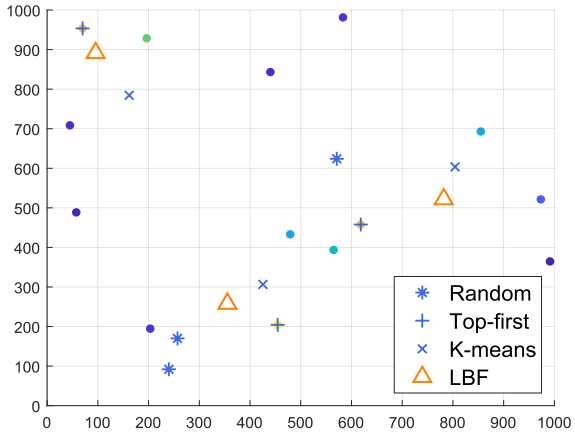

FIGURE 9. Server location of different algorithms in small scale simulation.

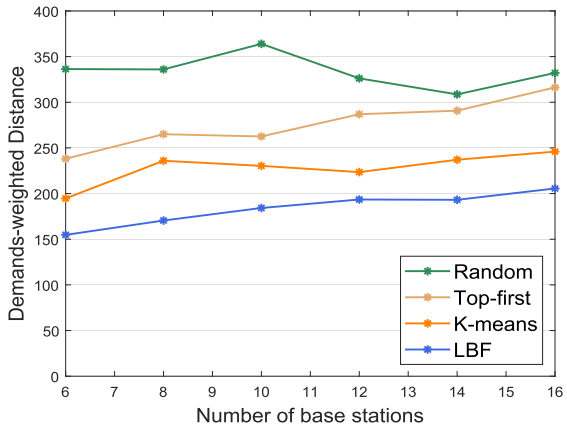

FIGURE 10. Variation of demands-weighted distance with the number of base stations in small scale simulation.

MEC server deployment on the condition of uncertain BSs demands.

TABLE 3. Demands-weighted distance vs number of BSs.

Number of BSs	Random	Top-First	K-means	LBF
6	336.338	238.124	194.655	154.759
8	335.947	265.000	235.921	170.484
10	363.955	262.528	230.320	184.211
12	326.144	286.878	223.487	193.484
14	308.617	290.779	237.041	193.149
16	332.101	316.234	245.931	205.582

D. LARGE SCALE SIMULATION

A large scale simulation is also conducted based on a real-world data set which consists of locations and users' access information of BSs from Shanghai Telecom [35]. The data set contains longitude and latitude data of 3233 BSs in Shanghai city and the corresponding users access information of each BS from June 1 to June 30, 2014.

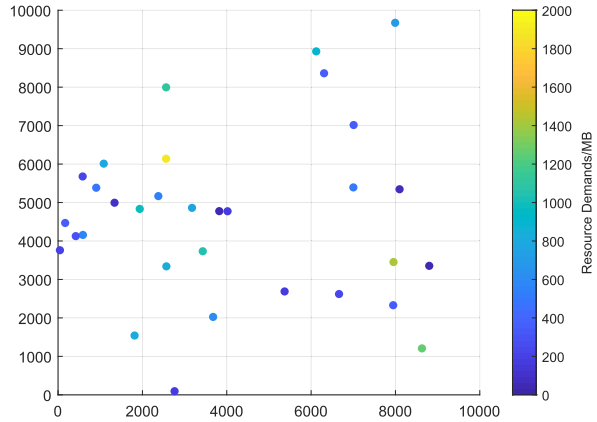

FIGURE 11. Base stations distribution and corresponding average demands in large scale simulation.

We choose 32 BSs from a region (Latitude:31.10∼31.22; Longtitude:121.11∼121.20) and map these BSs on a plane with coordinates by Millier Convertion. We generate traffic demands of each BS by collecting users access information for each day and assume that each access user has the same data requirements. The geographical distribution of these BSs and their corresponding average demands are shown in Figure 11. We utilized the data of the first 20 days as samples to train the neural network, and the last 10 days are adopted to evaluate the performance of the network. In addition, three edge servers were deployed in the system as in Section V-C. In Table 4 and Figure 12, we demonstrate the server locations of our learning-based framework (LBF) and other three benchmarking algorithms, whereas we show in Figure 13 the demands-weighted distance values achieved by these four schemes. As expected, in most of the time the performance ranking is LBF > K-means > Top-first > Random in terms of demands-weighted distance. Only on the 29th day, K-means is slightly better than our algorithm. We speculate that this is due to the different resource demands pattern

TABLE 4. Demands-weighted distance vs number of BSs.

Date	Random	Top-First	K-means	LBF
21	3907.85	2327.21	1905.11	1810.44
22	3608.91	2483.47	2048.34	1821.95
23	4019.41	2553.74	2043.20	1888.69
24	4187.74	3028.74	2246.00	1856.87
25	4123.03	3228.76	2332.77	1897.13
26	3833.35	2592.39	2079.40	1736.48
27	3842.26	2833.46	2140.00	1750.00
28	3638.15	2882.63	2036.33	2006.21
29	3830.15	2419.21	1948.49	2001.72
30	3740.86	2788.44	2135.76	1987.73

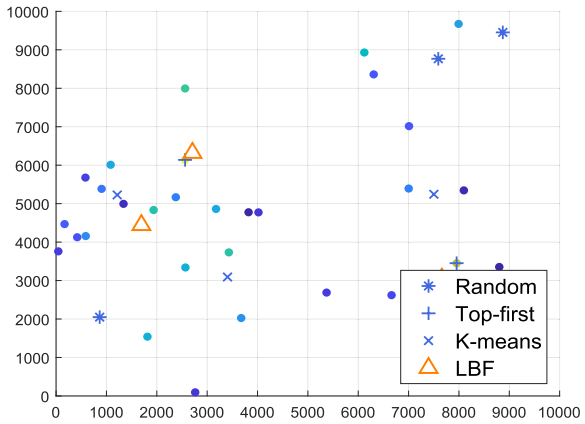

FIGURE 12. Server location of different algorithms in large scale simulation.

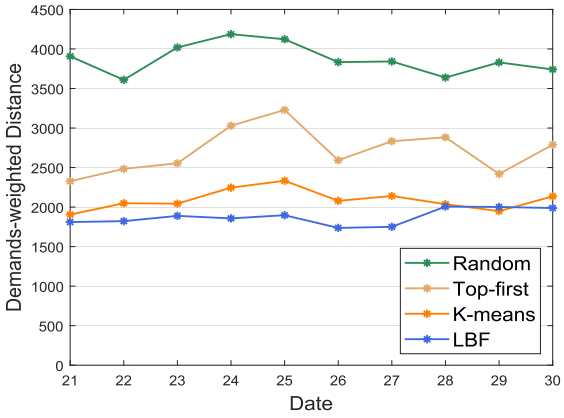

FIGURE 13. Variation of demands-weighted distance with the date in large scale simulation.

of that day. Consequently, the simulation results show that the learning-based framework is an efficient way to place edge servers in the actual network compared to other algorithms owing to its comprehensive considerations of uncertain BSs demands.

VI. CONCLUSION

As a key emerging technology, mobile edge computing can reduce latency and improve user experience by providing computing resources at the edge of the network. In this paper, we study the mobile edge server planning problem with the aim of minimizing the latency between edge server and BSs under the condition that the requests of BSs are arbitrary stochastic variables. We formulate the problem through a joint-uncertain problem formulation and propose a learning-based framework composed of stochastic simulation, neural network, and GA to practically solve the problem. We also discuss the practical relevance on utilizing the framework to determine edge server locations with historical BS demands data. Finally, we design experiments to evaluate the proposed framework by both generated BS demands and real BS demands data. The results show that the proposed framework is superior to other mechanisms in the two scenarios in terms of demands-weighted distance, which means our approach can find optimal locations for MEC servers when latency and uncertain BS demands are both considered. Moreover, the learning-based framework is essentially pragmatic as it can be applied on solving a group of problems with uncertain system parameters owing to its generality and universality. For future works, we aim to further investigate the MEC planning problem in various different scenarios, and applying deep learning models to improve the performance of the framework.

REFERENCES

[1] S. Wang, X. Zhang, J. Zhang, J. Feng, W. Wang, and K. Xin, "An approach for spatial-temporal traffic modeling in mobile cellular networks," in *Proc. 27th Int. Teletraffic Congr.*, Sep. 2015, pp. 203–209.

[2] S. Zhou, D. Lee, B. Leng, X. Zhou, H. Zhang, and Z. Niu, "On the spatial distribution of base stations and its relation to the traffic density in cellular networks," *IEEE Access*, vol. 3, pp. 998–1010, 2015.

[3] D. Lee, S. Zhou, X. Zhong, Z. Niu, X. Zhou, and H. Zhang, "Spatial modeling of the traffic density in cellular networks," *IEEE Wireless Commun.*, vol. 21, no. 1, pp. 80–88, Feb. 2014.

[4] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[5] Z. Li, X. Zhou, and Y. Qin, "A survey of mobile edge computing in the industrial Internet," in *Proc. 7th Int. Conf. Inf., Commun. Netw. (ICICN)*, Apr. 2019, pp. 94–98.

[6] S. M. A. Oteafy and H. S. Hassanein, "Leveraging tactile Internet cognizance and operation via IoT and edge technologies," *Proc. IEEE*, vol. 107, no. 2, pp. 364–375, Feb. 2019.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, pp. 1–16, Sep. 2015.

[8] B. Li, K. Wang, D. Xue, and Y. Pei, "K-means based edge server deployment algorithm for edge computing environments," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, pp. 1169–1174.

[9] 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.

[10] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 66–73.

[11] T. Lahderanta, T. Leppanen, L. Ruha, L. Loven, E. Harjula, M. Ylianttila, J. Riekki, and M. J. Sillanpaa, "Edge server placement with capacitated location allocation," 2019, *arXiv:1907.07349*. [Online]. Available: https://arxiv.org/abs/1907.07349

[12] K. Xiao, Z. Gao, Q. Wang, and Y. Yang, "A heuristic algorithm based on resource requirements forecasting for server placement in edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 354–355.

[13] G. Manasvi, A. Chakraborty, and B. S. Manoj, "Social network aware dynamic edge server placement for next-generation cellular networks," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 499–502.
[14] 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.*, early access, Feb. 24, 2020, doi: 10.1109/TII.2020.2975897.
[15] T.-L. Chin, Y.-S. Chen, and K.-Y. Lyu, "Queuing model based edge placement for work offloading in mobile cloud networks," *IEEE Access*, vol. 8, pp. 47295–47303, 2020.
[16] L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," *IEEE Trans. Services Comput.*, early access, Jan. 1, 2019, doi: 10.1109/TSC.2018.2890603.
[17] S. Lee, S. Lee, and M.-K. Shin, "Low cost MEC server placement and association in 5G networks," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2019, pp. 879–882.
[18] X. Xu, B. Shen, X. Yin, M. R. Khosravi, H. Wu, L. Qi, and S. Wan, "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Informat.*, early access, Apr. 16, 2020, doi: 10.1109/TII.2020.2987994.
[19] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE Netw.*, vol. 30, no. 3, pp. 22–29, May 2016.
[20] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.
[21] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1251–1275, 2nd Quart., 2020.
[22] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[23] L. Liu, Y. Cheng, L. Cai, S. Zhou, and Z. Niu, "Deep learning based optimization in wireless network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[24] B. Liu, *Theory and Practice of Uncertain Programming*, vol. 239. Berlin, Germany: Springer-Verlag, Jan. 2009.
[25] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 693–701.
[26] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2020.
[27] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1468–1476.
[28] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, early access, Oct. 7, 2019, doi: 10.1109/TMC.2019.2945956.
[29] A. Laghrissi, T. Taleb, and M. Bagaa, "Conformal mapping for optimal network slice planning based on canonical domains," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 519–528, Mar. 2018.
[30] 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.
[31] Y. Sun, X. Chen, D. Liu, and Y. Tan, "Power-aware virtual machine placement for mobile edge computing," in *Proc. Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2019, pp. 595–600.
[32] W. Wang, Y. Zhao, M. Tornatore, A. Gupta, J. Zhang, and B. Mukherjee, "Virtual machine placement and workload assignment for mobile edge computing," in *Proc. IEEE 6th Int. Conf. Cloud Netw. (CloudNet)*, Sep. 2017, pp. 1–6.
[33] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1031–1045, Apr. 2017.
[34] J. Meng, C. Zeng, H. Tan, Z. Li, B. Li, and X.-Y. Li, "Joint heterogeneous server placement and application configuration in edge computing," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 488–497.
[35] Telecom, Shanghai, China. (2020). *The Distribution of 3233 Base Stations*. [Online]. Available: http://www.sguangwang.com/dataset/telecom.zip
[36] J. G. Andrews, F. Baccelli, and R. K. Ganti, "A tractable approach to coverage and rate in cellular networks," *IEEE Trans. Commun.*, vol. 59, no. 11, pp. 3122–3134, Nov. 2011.
[37] H. S. Dhillon, R. K. Ganti, F. Baccelli, and J. G. Andrews, "Modeling and analysis of K-Tier downlink heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 3, pp. 550–560, Apr. 2012.

MENGHAN SHAO received the B.S. degree in communication engineering from Xidian University, Xi'an, China, in 2018, where she is currently pursuing the M.S. degree in electronics and communication with the State Key Laboratory of Integrated Service Networks. Her research interests include 5G networks and mobile edge computing.

JIAYI LIU (Member, IEEE) received the B.Sc. degree in electronics engineering from Xidian University, Xi'an, China, in 2007, the M.Sc. degree in computer science from Télé in 2009, and the Ph.D. degree in computer science from Rennes 1 University, France, in 2013. Since 2014, she has been a Lecturer with the School of Telecommunication Engineering, Xidian University, and the Guangzhou Institute of Technology. Her research interests include 5G networks, content distribution, and network resource scheduling.

QINGHAI YANG (Member, IEEE) received the B.S. degree in communication engineering from the Shandong University of Technology, China, in 1998, the M.S. degree in information and communication systems from Xidian University, China, in 2001, and the Ph.D. degree in communication engineering from Inha University, South Korea, in 2007. From 2007 to 2008, he was a Research Fellow of the UWB-ITRC, South Korea. Since 2008, he has been a Full Professor with Xidian University. His current research interests include the fields of autonomic communication, content delivery networks, and LTE-A techniques. He received the University-President Award for the Ph.D. degree.

GWENDAL SIMON (Member, IEEE) is a Research Expert of Network Measurement with Huawei Technologies Company Ltd. Previously, he was a Full Professor at IMT Atlantique from 2006 to 2019. He was a Visiting Researcher at the University of Waterloo from 2011 to 2012 and a Visiting Senior Researcher at Adobe Research, San Jose, CA, USA, from 2018 to 2019. He was a Research Engineer at Orange Research and Development Labs from 2001 to 2006. He has coauthored over 100 conference papers/journal articles in networks, multimedia (video, gaming, and virtual reality), and systems. He received several best paper awards (including the IEEE ICC and ACM MMSys).