Cost-Efficient Server Configuration and Placement for Mobile Edge Computing

Zhenli He, Kenli Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

Abstract—Computing resource configuration and site selection of edge servers (ESs) are two critical steps to build up a mobile edge computing (MEC) platform. In this paper, the joint optimization problem of configuration and placement for ES in the MEC environment is investigated. First, we treat each ES as an M/G/m queueing model, and establish mathematical models to characterize the MEC environment, such that the performance and operational expenditures (OPEX) of the system can be calculated analytically. Then, we design a two-stage method and develop a series of algorithms based on bisection algorithm and genetic algorithm (GA) to obtain the optimal configuration scheme and sub-optimal placement scheme (including the deployment quantity) of ESs, with the goal of minimizing OPEX while maintaining system performance at a predetermined level. Finally, we conduct experiments based on a real base station dataset provided by Shanghai Telecom to show the effectiveness of the proposed algorithms. To the best of our knowledge, this work is the first research of the joint optimization problem of configuration and placement for ES in the MEC environment, where the main objective is to increase the cost efficiency.

Index Terms—Cost efficiency, configuration scheme, edge server, mobile edge computing, placement scheme

1 Introduction

1.1 Motivation

According to a recent forecast by Cisco [1], the number of global mobile users will grow from 5.1 billion in 2018 to 5.7 billion by 2023, and the mobile data traffic will grow at an annual rate of 57%. By 2023, 299.1 billion mobile applications will be downloaded globally, and the new generation mobile applications with complex computing needs (e.g., large-scale image processing, personal assistant) will be widely used. Despite the fact that mobile devices (MDs) are becoming more powerful and intelligent, the rise of new generation applications and the increasing demand for portable services pose significant challenges to the global mobile network environment [2].

- *Zhenli He is with the School of Software of Yunnan University, Kunming, Yunnan 650504, China, and with the School of Computer Science and Electronic Engineering of Hunan University, Hunan 410082, China, and also with the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China. E-mail: hezl@ynu.edu.cn.*
- *Kenli Li is with the School of Computer Science and Electronic Engineering of Hunan University, Hunan 410082, China, and also with the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China. E-mail: lkl@hnu.edu.cn.*
- *Keqin Li is with the School of Computer Science and Electronic Engineering of Hunan University, Hunan 410082, China, and with the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 7 May 2021; revised 5 Nov. 2021; accepted 13 Dec. 2021. Date of publication 16 Dec. 2021; date of current version 14 Feb. 2022.
This work was supported in part by the National Natural Science Foundation of China under Grant 61876061, in part by the Applied Basic Research Foundation of Yunnan Province under Grant 202001BB050034, and in part by the Open Foundation of Key Laboratory in Software Engineering of Yunnan Province under Grant 2020SE405.
(Corresponding authors: Keqin Li and Kenli Li.)
Recommended for acceptance by D. Talia.
Digital Object Identifier no. 10.1109/TPDS.2021.3135955

As a promising computing paradigm, mobile edge computing (MEC) offloads computation-intensive tasks from MDs with limited computing capability and battery power to edge servers (ESs) that are closer to users. It not only relieves the pressure on MDs and backbone networks, but also avoids the problem of high-latency communication, and will become the mainstream computing paradigm. According to International Data Corporation (IDC) prediction [3], global data will reach 180 zettabytes, of which more than 70% of the data will be analyzed, processed and stored on the edge of the network by 2025.

The first step in constructing a MEC platform is to deploy ESs, which raises the site selection problem for ESs. However, this problem is different from the traditional facility location problem [4], as the placement scheme of ESs is coupled with the resource configuration scheme of ESs, and both of them will affect the operational expenditures (OPEX) and system performance [5]. As a result, the joint optimization problem of configuration and placement for ESs needs to be solved before building up MEC systems.

High-density deployment and excessive computing resource configuration of ESs will directly reduce the Return On Investment (ROI) of mobile network operator (MNO), whereas insufficient deployment density and resource configuration will reduce the Quality of Service (QoS) and Quality Of Experience (QoE). In general, concentrating computing resources at a few large edge nodes can help MNO save OPEX, but this comes at the price of potential degradation of QoS and QoE. It is necessary not only to ensure the QoS and QoE of MEC platform, but also to control the OPEX of system, which poses a significant challenge to the solution of this problem.

1.2 Our Contributions

In this paper, we consider a typical application scenario from the standpoint of an MNO, in which we will deploy

some ESs at some base stations (BSs) to establish an MEC platform. Obviously, if we deploy ESs with abundant computing resources at all BSs, our users will have an unparalleled quality of experience. However, this plan is unrealistic, because it will result in a dramatic explosion of our budget and will fail due to high costs and low returns. We not only need to find the strategic deployment locations of ESs, but also need to obtain the optimal computing resource configuration for each ES, such that we can provide a certain service-quality guarantee after the system is established while minimizing OPEX to increase the cost efficiency.

Our major contributions in this paper can be summarized as follows.

- We treat each ES as an M/G/m queueing model, and establish mathematical models to characterize the MEC environment, such that the performance and OPEX of the MEC platform can be calculated analytically.
- We formulate the cost-efficient ES configuration and placement problem as a NP-hard combinatorial optimization problem with three constraints, namely maximum configuration constraint, computation capacity constraint, and performance constraint.
- We design a two-stage method and develop a series of algorithms based on bisection algorithm and genetic algorithm (GA) to obtain the optimal configuration scheme and sub-optimal placement scheme (including the deployment quantity) of ESs, such that the OPEX of system is minimized under the premise that the system performance reaches the predetermined standard.
- We also conduct experiments based on a real base station dataset provided by Shanghai Telecom to show the effectiveness of the proposed algorithms.

To the best of our knowledge, this work is the first research of the joint optimization problem of configuration and placement for ES in the MEC environment, where the main objective is to increase the cost efficiency. The rest of the paper is organized as follows. In Section 2, we review related research. In Section 3, we introduce the needed preliminaries. In Section 4, we formulate the cost-efficient ES configuration and placement problem. In Section 5, we propose our solution for the problem, while in Section 6, we illustrate the comparative experimental evaluation results. In Section 7, we conclude this paper.

2 Related Research

As the computing resource configuration and site selection of ESs are the critical steps to build up an MEC platform, research into these issues has grown in popularity. However, most of the existing research considers these two issues separately, ignoring the coupling between the configuration scheme and placement scheme. On the other hand, there is a type of problem in the existing research that at first glance is similar to the joint optimization problem of configuration and placement for ES, that is, joint resource allocation and service placement problem in the context of network function virtualization (NFV). In this section, we first review the existing research from three aspects, namely, edge server configuration, edge server placement, and resource allocation and service placement in the context of NFV, and then discuss the differences between our research and other existing research.

Edge Server Configuration. A common method to improve cost efficiency is dynamic scaling mechanism [6], that is, dynamically shutting down server instances or reducing CPU-cycle frequencies of ESs based on the real-time workloads. In [7], the authors studied the cost-efficient resource provisioning problem of ESs. They established an M/M/1 queueing model to characterize ESs, and proposed an algorithm to dynamically adjust the computation capacity of ESs and cloud instance. In [8], the authors proposed a flexible resource distribution scheme for IoT devices to optimize the execution cost and fault-tolerance. Mao *et al.* [9] designed an algorithm to dynamically adjust CPU-cycle frequencies of ESs and obtain the optimal task scheduling decision according to the computation demands and wireless fading channel. In [10], the authors designed a dynamic microservice scheduling scheme for an MEC-enabled IoT platform to provide fair QoS and satisfaction-level to IoT devices. Wang *et al.* [11] studied the joint optimization of computing offloading, resource allocation and caching strategy. Samanta *et al.* [12] proposed an adaptive service scheduling scheme to improve the resource utilization while minimizing total service latency. In [13], the authors designed a dynamic resource allocation mechanism to optimize the profit and economic balance of edge clouds. However, the application premise of dynamic scaling mechanism is that the computing resources configured on the ES are sufficient, that is, the ES can meet the computation demands of the area in which it is located. It is expensive to configure all ESs with maximum computing resources. Such a configuration scheme is irrational for the ESs located in suburbs with lower computation demands. Therefore, the most fundamental solution remains optimizing the resource configuration of ESs in advance according to computation demands. In our previous research work [14], we studied the optimization of server configuration in the MEC environment. We established an M/M/m queueing model to characterize ESs, and designed a series of fast numerical algorithms to determine the optimal number of processors that should be configured on each ES.

Edge Server Placement. In recent years, the placement of ESs has received more and more attention. In [15], the authors studied the joint ES placement and task scheduling problem by mixed integer programming (MIP). Mondal *et al.* [16] studied the ES placement problem in passive optical networks, and constructed a nonlinear MIP to determine the optimal location of ESs to minimize the deployment cost. In [17], the authors proposed an ES placement strategy for the purpose of minimizing end-to-end delay. Wang *et al.* [18] formulated the ES placement as a multi-objective constraint optimization problem, and adopted MIP to find the optimal deployment locations of K ESs. In [19], the authors formulated ES placement problem as a capacitated facility location problem, and adopted MIP to determine the deployment location of K ESs. In addition to the method of MIP, there are some existing research adopted clustering algorithm to obtain ES placement scheme. In [20], the

authors studied the ES placement problem in wireless metropolitan area networks. They designed an algorithm to obtain multiple ES placement schemes according to the computation demands snapshot of multiple time periods, and obtained the final placement scheme by using K-medians clustering algorithm. Similar work can be found in [21]. There are also some works that use evolutionary algorithms to obtain ES placement scheme. In [22], the authors designed an ES placement strategy based on particle swarm optimization to find the optimal deployment locations of ESs. Xu *et al.* [23] formulated ES placement problem as a multi-objective optimization problem, then designed a strategy based on genetic algorithm to obtain the placement scheme with low latency and balanced workload. However, all these studies assume that the computing resources configured on ESs are the same, without considering the heterogeneity, and ignoring the dependence of placement scheme on resource configuration scheme. On the other hand, some studies assume that the deployment quantity of ESs is known, but it is difficult to directly define an optimal deployment quantity in practical applications.

Resource Allocation and Service Placement in the context of NFV. With the development of network function virtualization technology (NFV), some scholars have pointed out that MEC applications can be deployed in a NFV environment, which raises the resource allocation and service placement problem in the context of NFV. In [24], the authors proposed a service placement algorithm applied in an MEC-NFV environment, where the main objective is to maximize service availability while minimizing latency. Ma *et al.* [25] studied the joint virtualized network function (VNF) instance placement and user assignment problem in an MEC-NFV environment, where the main objective is to maximize the number of served users while minimizing the admission cost. In [26], the authors proposed an algorithm that dynamically allocates VNF instances to MEC or cloud data centers, where the main objective is to maximize the number of served users. Wang *et al.* [27] studied the service placement problem in an MEC-NFV environment, and constructed a Hungarian-based algorithm to optimize resource utilization and QoS. In [28], the authors studied the joint VNF placement and resource allocation problem in an MEC-NFV environment by MIP and GA-based method, where the main objective is to minimize the management cost.

In order to clearly position our investigation and highlight our unique features, we analyze the differences between our research and above existing research.

- Our study is different from the above-mentioned research on edge server configuration and edge server placement in that we consider the coupling between the configuration scheme and placement scheme. In this paper, we design a series of algorithms to simultaneously obtain the sub-optimal deployment quantity and locations of ESs, as well as the optimal number and speed of processors configured for each ES.
- The prerequisites considered in our research are different from that considered in joint resource allocation and service placement problem. In joint resource allocation and service placement problem, the ESs have been deployed to provide services for different MDs, that is, the total amount and type of computing resources within each ES are determined and known. In our study, the ESs are not yet deployed, that is, the MNO want to deploy ESs at the strategic deployment locations to build an MEC platform.
- The optimization objective considered in our research is different from that considered in joint resource allocation and service placement problem. The optimization objective of joint resource allocation and service placement problem is usually to improve service availability or maximize the number of served users while reducing management or admission cost. The main objective of our study is to provide a certain service-quality guarantee after the system is established while minimizing OPEX to increase the cost efficiency.

3 Preliminaries

In this section, we introduce the needed preliminaries, including assumptions, notations, and models that will be used throughout the rest of the paper. (For reader's convenience, Section 1 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2021.3135955 gives a summary of notations and their definitions in the order introduced in the paper.)

3.1 The MEC Environment

First, we introduce some key characteristics of the MEC environment discussed in this paper.

Assume that there are n BSs (which denoted as $B = \{b_1, b_2, \ldots, b_n\}$) deployed at n strategic locations (which denoted as $L = \{l_1, l_2, \ldots, l_n\}$) by network operators to provide wireless access network for MDs within their coverage (i.e., the wireless coverage problems have been considered). The deployment location of the jth BS (i.e., b_j) can be denoted as

$$l_j \triangleq (x_j, y_j),\ 1 \leq j \leq n, \tag{1}$$

where x_j and y_j respectively represent the latitude and longitude of the location l_j. We also assume that these n BSs are interconnected through Metropolitan Area Network (MAN), which is an existing interconnection network infrastructure.

In order to establish the MEC platform, we will strategically deploy some ESs at the locations of some suitable BSs, and then configure appropriate computing resources for each ES. Thus, we need to make decisions on the deployment quantity and locations of ESs, as well as the ES configuration scheme. Let k denote the deployment quantity of ESs, $S = \{s_1, s_2, \ldots, s_k\}$ denote the set of ESs that need to be deployed, and $\hat{L} = \{\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_k\}$ denote the deployment locations of k ESs. The deployment location of the ith ES (i.e., s_i) can be expressed as

$$\hat{l}_i \triangleq (\hat{x}_i, \hat{y}_i),\ 1 \leq i \leq k, \tag{2}$$

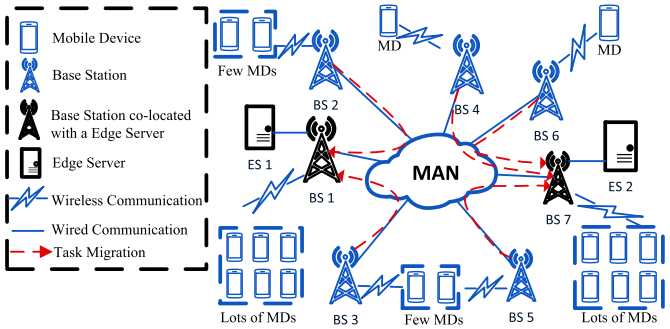

Fig. 1. An example of the MEC Environment.

where $\hat{x}_i$ and $\hat{y}_i$ respectively represent the latitude and longitude of the location $\hat{l}_i$. Then, we can obtain the spherical distance between BS b_j and ES s_i by calculating

$$d(l_j, \hat{l}_i) = 2R \cdot \arcsin\left(\sqrt{hav(\hat{x}_i - x_j) + \cos(x_j)\cos(\hat{x}_i)hav(\hat{y}_i - y_j)}\right), \quad (3)$$

where $R = 6371.009$ kilometers denotes the mean earth radius and hav is the Haversine function (which can be calculated by $hav(\theta) = \sin^2(\theta/2)$). It should be noted that the deployment quantity k and the deployment locations $\hat{L}$ are not parameters determined in advance, but the solutions we need to obtain, which will be discussed in detail in Section 4.

Since the set of ES deployment locations is actually a proper subset of the set of BS deployment locations (i.e., $\hat{L} \subset L$), for every $\hat{l}_i$ in $\hat{L}$, there is exactly one element l_j from L that equal to $\hat{l}_i$. We use a function h to represent the relation, i.e., $h(i) = j$, which represents the ES s_i will be deployed at the BS b_j. Notice that the mapping relation depends on the ES placement scheme.

If the ES s_i is co-located with the BS b_j, the MDs within b_j's coverage will directly offload computation-intensive tasks (i.e., offloadable tasks) to s_i through wireless access network; otherwise, the BS b_j needs to further relay the offloadable tasks to the nearest ES via MAN and resulting in an extra transmission latency. For example, Fig. 1 indicates a small MEC environment composed of widely distributed MDs, seven BSs (represented by BSs 1∼7), and two ESs (represented by ESs 1∼2). The MDs transmit their offloadable tasks to the BSs through wireless access network (represented by the lightning connections). For BSs 1 and 7, the offloadable tasks will be directly processed by Edge Server 1 and 2, respectively. For BSs 2∼6, the offloadable tasks will be migrated to the nearest ES for processing (represented by the red dotted arrows).

We assume that the stable computation demands of all BSs are known. Although the computation demands of BSs are different and dynamically changed due to their different geographies and the mobility of MDs, we think that a time-series analysis based on the historical mobile data traffic can be used to predict the relatively stable computation demands. It should be noted that the ES configuration and placement decisions only need to be made once before system deployment, and we can make new decisions when the environment changes significantly. In order to quantify the stable computation demands, we also assume that each BS b_j accepts a Poisson stream of computation tasks with arrival rate λ_j (measured by the average number of arriving tasks in one unit time), which is based on the fact that the occurrence of many things in nature obeys the Poisson distribution, and the Poisson distribution can also estimate many other probability distributions.

Based on the above descriptions, we further mathematically describe the BSs and ESs. In this paper, each BS b_j is denoted as

$$b_j \triangleq (l_j, \lambda_j),\ 1 \le j \le n,\ l_j \in L, \quad (4)$$

and each ES s_i is denoted as

$$s_i \triangleq (\hat{l}_i, m_i, f_i, \hat{\lambda}_i),\ 1 \le i \le k,\ \hat{l}_i = l_{h(i)} \in L, \quad (5)$$

where $\hat{\lambda}_i$ represent total computation demands of ES s_i, and we will configure m_i processors with same execution speed f_i (which is measured in units of billion instructions per second, i.e., BIPS) for ES s_i. Since ES s_i not only needs to accept the offloadable tasks from the BS which deployed at location $\hat{l}_i$ (i.e., $l_{h(i)}$), but also needs to accept the offloadable tasks relay by other BSs, the value of $\hat{\lambda}_i$ can be calculated by

$$\hat{\lambda}_i = \lambda_{h(i)} + \sum\nolimits_{l_v \in ne(i)} \lambda_v, \quad (6)$$

where $ne(i)$ denotes a subset of BS deployment locations, and for each l_v in $ne(i)$, we have

$$\begin{cases} l_v \in L - \hat{L}, \\ d(l_v, \hat{l}_i) < d(l_v, \hat{l}_w),\ \forall \hat{l}_w \in \hat{L}\ and\ \hat{l}_w \ne \hat{l}_i. \end{cases} \quad (7)$$

As ES have limited resources compared to cloud data center, we use $m_{\max}$ and $f_{\max}$ to represent the maximum number and maximum speed of processors configured for an edge server, respectively. Then, we have $m_i \le m_{\max}$ and $f_i \le f_{\max}$, for all $1 \le i \le k$.

We use $u_j \in \{0, 1\}$ to indicate whether an ES will be placed at BS b_j (i.e., location l_j), and $u_j = 1$ means that an ES is placed at b_j, otherwise, $u_j = 0$. Then we have

$$k = \sum_{j=1}^{n} u_j. \quad (8)$$

The vector $(u_1, u_2, \ldots, u_n)$ is actually a placement scheme of ESs, and the vector $(m_1, f_1, m_2, f_2, \ldots, m_k, f_k)$ is actually a configuration scheme of ESs.

3.2 Average Response Time of Tasks

In order to ensure that the platform has a certain service-quality guarantee after establishment, we need to establish mathematical models to analyze the average response time of all offloadable tasks in the environment. Generally, network latency mainly consists of four parts, namely, transmission latency, propagation latency, processing latency and queueing latency. However, the propagation latency is usually considered negligible compared to the time required to transmit the packet. Thus, we only focus on network latency caused by transmission, processing, and queuing.

First, we define some symbols to characterize the offloadable tasks. The execution requirements (measured in units of billion instructions, namely, BI) of offloadable tasks on ES

s_i are independent and identically distributed (i.i.d.) random variables r_i with the mean $\overline{r_i}$ and second moment $\overline{r_i^2}$. The sizes of computation input data (measured in units of million bits, i.e., Mb) involved in offloadable tasks are i.i.d. random variables d_i with the mean $\overline{d_i}$ and second moment $\overline{d_i^2}$. The execution requirements and input data sizes can obey arbitrary probability distribution. The values of $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, and $\overline{d_i^2}$ can be obtained by the methods of graph analysis in [29] and [30].

Second, we define some symbols to characterize the two data transmission rates involved in this paper. The wireless data transmission rate (measured in units of million bits per second, i.e., Mbps) in the wireless transfer channel between MDs (within s_i's coverage) and ES s_i are i.i.d. random variables $\dot{c}_i$ with an arbitrary probability distribution. We assume that its mean $\overline{\dot{c}_i}$ and second moment $\overline{\dot{c}_i^2}$ are available. The wired data transmission rate between BSs (nearest to s_i but without an ES) and ES s_i in MAN are i.i.d. random variables $\ddot{c}_i$ with an arbitrary probability distribution. We also assume that its mean $\overline{\ddot{c}_i}$ and second moment $\overline{\ddot{c}_i^2}$ are available.

Third, we use an M/G/m queueing model to characterize multiple heterogeneous edge servers (since ESs will be configured with different numbers of processors, and the processors on different ES may be different types and therefore have different execution speeds) to conduct a rigorous analysis of network latency. The execution times of the tasks offloaded from the MDs within $b_{h(i)}$'s coverage to the ES s_i are i.i.d random variables $r_i/f_i + d_i/\dot{c}_i$, where r_i/f_i is the processing latency and $d_i/\dot{c}_i$ is the wireless transmission latency. The random variables have mean $\overline{r_i}/f_i + \overline{d_i}/\overline{\dot{c}_i}$ and second moment $\overline{r_i^2}/f_i^2 + 2\overline{r_i}\,\overline{d_i}/(f_i\overline{\dot{c}_i}) + \overline{d_i^2}/\overline{\dot{c}_i^2}$. The execution times of the tasks relayed from other BSs to the ES s_i are i.i.d random variables $r_i/f_i + d_i/\dot{c}_i + d_i/\ddot{c}_i$, where r_i/f_i is the processing latency, $d_i/\dot{c}_i$ is the wireless transmission latency, and $d_i/\ddot{c}_i$ is the transmission latency caused by task migration. The random variables have mean $\overline{r_i}/f_i + \overline{d_i}/\overline{\dot{c}_i} + \overline{d_i}/\overline{\ddot{c}_i}$ and second moment $\overline{r_i^2}/f_i^2 + \overline{d_i^2}/\overline{\dot{c}_i^2} + \overline{d_i^2}/\overline{\ddot{c}_i^2} + 2\overline{r_i}\,\overline{d_i}/(f_i\overline{\dot{c}_i}) + 2\overline{r_i}\,\overline{d_i}/(f_i\overline{\ddot{c}_i}) + 2\overline{d_i}^2/(\overline{\dot{c}_i}\,\overline{\ddot{c}_i})$. Therefore, the execution times of all tasks on ES s_i are i.i.d. random variables with mean

$$\overline{t_i} = \frac{\lambda_{h(i)}}{\hat{\lambda}_i}\left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\overline{\dot{c}_i}}\right) + \frac{\sum_{l_v \in ne(i)} \lambda_v}{\hat{\lambda}_i}\left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\overline{\dot{c}_i}} + \frac{\overline{d_i}}{\overline{\ddot{c}_i}}\right), \tag{9}$$

and the second moment

$$\overline{t_i^2} = \frac{\lambda_{h(i)}}{\hat{\lambda}_i}\left(\frac{\overline{r_i^2}}{f_i^2} + \frac{\overline{d_i^2}}{\overline{\dot{c}_i^2}} + \frac{2\overline{r_i}\,\overline{d_i}}{f_i\overline{\dot{c}_i}}\right) + \frac{\sum_{l_v \in ne(i)} \lambda_v}{\hat{\lambda}_i}\left(\frac{\overline{r_i^2}}{f_i^2} + \frac{\overline{d_i^2}}{\overline{\dot{c}_i^2}} + \frac{\overline{d_i^2}}{\overline{\ddot{c}_i^2}} + \frac{2\overline{r_i}\,\overline{d_i}}{f_i\overline{\dot{c}_i}} + \frac{2\overline{r_i}\,\overline{d_i}}{f_i\overline{\ddot{c}_i}} + \frac{2\overline{d_i}^2}{\overline{\dot{c}_i}\,\overline{\ddot{c}_i}}\right), \tag{10}$$

and the variance

$$\sigma_i^2 = \overline{t_i^2} - \overline{t_i}^2, \tag{11}$$

and the coefficient of variation

$$CV_i = \frac{\sigma_i}{\overline{t_i}} = \sqrt{\frac{\overline{t_i^2}}{\overline{t_i}^2} - 1}, \tag{12}$$

where the $\lambda_{h(i)}/\hat{\lambda}_i$ is the percentage of tasks offloaded from the MDs within $b_{h(i)}$'s coverage to the ES s_i and $\sum_{l_v \in ne(i)} \lambda_v/\hat{\lambda}_i$ is the percentage of tasks relayed from other BSs to the ES s_i.

According to queuing theory, the utilization of ES s_i is $\rho_i = (\hat{\lambda}_i \overline{t_i})/m_i$, that is,

$$\rho_i = \frac{\lambda_{h(i)}}{m_i}\left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\overline{\dot{c}_i}}\right) + \frac{\sum_{l_v \in ne(i)} \lambda_v}{m_i}\left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\overline{\dot{c}_i}} + \frac{\overline{d_i}}{\overline{\ddot{c}_i}}\right). \tag{13}$$

Based on the accurate approximation of average queueing latency in an M/G/m queueing system from the work [31], we can get the average queueing latency of the tasks on ES s_i as

$$\overline{w_i} = \left(\frac{CV_i^2 + 1}{2}\right)\overline{w_i}^*, \tag{14}$$

where $\overline{w_i}^*$ is the average queueing latency of all tasks in an M/M/m queueing system with the same utilization as the M/G/m queueing system. And we have [32, p. 102]

$$\overline{w_i}^* = \overline{t_i}\,\frac{p_{i,m_i}}{m_i(1-\rho_i)^2}, \tag{15}$$

where

$$p_{i,m_i} = p_{i,0}\,\frac{(m_i\rho_i)^{m_i}}{m_i!}, \tag{16}$$

and

$$p_{i,0} = \left(\sum_{l=0}^{m_i-1}\frac{(m_i\rho_i)^l}{l!} + \frac{(m_i\rho_i)^{m_i}}{m_i!}\cdot\frac{1}{1-\rho_i}\right)^{-1}. \tag{17}$$

Therefore, the *average response time* of all offloadable tasks in the environment is

$$\overline{T} = \sum_{i=1}^{k}\frac{\hat{\lambda}_i}{\lambda}(\overline{t_i} + \overline{w_i}), \tag{18}$$

where $\lambda = \hat{\lambda}_1 + \hat{\lambda}_2 + \cdots \hat{\lambda}_k = \lambda_1 + \lambda_2 + \cdots \lambda_n$ denotes the total computation demands in the environment.

3.3 OPEX of MEC Platform

In this section, we establish mathematical models to analyze the OPEX of the MEC platform.

The OPEX depends on two main factors, namely, site rentals and energy consumption cost. The site rentals primarily refer to the costs associated with renting sites for ES deployment, and they are various in terms of different geographies. The energy consumption cost primarily refer to the cost of providing power to ESs, which is proportional to the number of processors configured on ESs and their computing energy consumption.

Let c_j represent the annual site rental (measured in units of CNY/year) of the deployment location l_j (which can be estimated based on the average annual rental of the surrounding houses), and κ denotes the economic lifecycle (measured in units of years) of the MEC platform. Then, the total site rentals during the economic lifecycle can be calculated by

$$C_s = \kappa\sum_{j=1}^{n} u_j c_j. \tag{19}$$

The processor power consumption is the dominant component of ES's energy consumption, which is typically represented as

$$P_i = \xi {f_i}^{\alpha}, \tag{20}$$

where f_i is the processor execution speed, and ξ and α are technology-dependent constants [33], [34]. We formulate the average power consumption of ES s_i (measured in units of Watts/second) as

$$P_i = m_i(\rho_i \xi {f_i}^{\alpha} + P^*), \tag{21}$$

where ρ_i is the utilization of ES s_i, and P^* denotes the base power of the processor including static and short-circuits power dissipation. Then, we can get the total energy consumption cost during the economic lifecycle as

$$C_p = \kappa \upsilon C_e \sum_{i=1}^{k} P_i = \kappa \upsilon C_e \sum_{i=1}^{k} (m_i(\rho_i \xi {f_i}^{\alpha} + P^*)), \tag{22}$$

where $\upsilon = 31536000$ is a constant denotes the total number of seconds in one year, i.e., 365 days $\cdot$ 24 hours $\cdot$ 60 minutes $\cdot$ 60 seconds, and C_e represents the price of electricity per Watt per second (measured in units of CNY).

Based on the above discussion, the OPEX of the MEC platform is

$$C = C_s + C_p = \kappa \sum_{j=1}^{n} u_j c_j + \kappa \upsilon C_e \sum_{i=1}^{k} (m_i(\rho_i \xi {f_i}^{\alpha} + P^*)). \tag{23}$$

4 Problem Definition

Based on the notations and models above, we know that both the OPEX and system performance are closely related to the configuration scheme and placement scheme of ESs. Generally, deploying more ESs with sufficient computing resources can achieve better performance, but this comes at the price of increased OPEX. Thus, optimizing both performance and OPEX may be conflicting requirements, and we need to make a trade-off between them. In this paper, we consider minimizing the OPEX of the MEC platform while ensuring that the system performance reaches the predetermined standard, thereby increasing the cost efficiency. The cost-efficient ES configuration and placement problem can be mathematically formulated as follows.

Given n BSs $b_1, b_2, \ldots, b_n$, the deployment locations of BSs $l_1, l_2, \ldots, l_n$, the annual site rentals $c_1, c_2, \ldots, c_n$, the task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, the transmission related parameters including $\overline{\dot{c}_i}$, $\overline{{\dot{c}_i}^2}$, $\overline{\ddot{c}_i}$, and $\overline{{\ddot{c}_i}^2}$, for all $1 \leq i \leq k$, the task related parameters including $\overline{r_i}$, $\overline{{r_i}^2}$, $\overline{d_i}$, and $\overline{{d_i}^2}$, for all $1 \leq i \leq k$, the energy consumption parameters including ξ, α, P^*, and C_e, and the economic lifecycle κ, find the optimal ES configuration scheme $(m_1, f_1, m_2, f_2, \ldots, m_k, f_k)$ and the sub-optimal ES placement scheme $(u_1, u_2, \ldots, u_n)$ (notice that $k = \sum_{j=1}^{n} u_j$ is the deployment quantity of ESs), such that the OPEX of the MEC platform is minimized, namely,

$$minimize \;\; C = \kappa \sum_{j=1}^{n} u_j c_j + \kappa \upsilon C_e \sum_{i=1}^{k} (m_i(\rho_i \xi {f_i}^{\alpha} + P^*)),$$

subject to the following constraints

$$u_j \in \{0, 1\}, \tag{24}$$

$$m_i \leq m_{\max}, \text{ for all } 1 \leq i \leq k, \tag{25}$$

$$f_i \leq f_{\max}, \text{ for all } 1 \leq i \leq k, \tag{26}$$

$$\rho_i < 1, \text{ for all } 1 \leq i \leq k, \tag{27}$$

$$\overline{T} = \tilde{T}, \tag{28}$$

where constraints (25) and (26) jointly ensure that the number and speed of processors configured on each ES do not exceed the maximum configuration (namely, maximum configuration constraint), constraint (27) ensures that the workload of each ES do not exceed its computation capacity (namely, computation capacity constraint), and constraint (28) ensures that the system performance must be close to the given performance constraint $\tilde{T}$. It should be noted that the value of $\tilde{T}$ should be determined by the MNO according to specific user requirements or company plans.

5 Our Solutions

In this section, we conduct some preliminary analysis of the cost-efficient ES configuration and placement problem and then develop a series of algorithms to solve it.

5.1 Analysis

It is clear that the average task response time and OPEX of the MEC platform are not only related to the placement scheme of ESs, but also related to the configuration scheme of ESs. However, it is extremely difficult and challenging to obtain the two optimal schemes at the same time. Therefore, we attempt to decouple these dependencies.

Based on the problem definition in Section 4, the configuration scheme does not affect the site rentals C_s, but only affects the energy consumption cost C_p. Given an ES placement scheme, if we can obtain the optimal configuration scheme of ESs, the energy consumption cost C_p can be regarded as a function of the ES placement scheme. Then, the problem we need to solve only depends on the locations of ESs. Therefore, we design a two-stage method, which is described as follows.

5.1.1 Stage I

Given an ES placement scheme (i.e., the task arrival rates $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_k$, λ, the deployment locations of ESs and the number of ESs k have been determined), and the related parameters including $\overline{\dot{c}_i}$, $\overline{{\dot{c}_i}^2}$, $\overline{\ddot{c}_i}$, $\overline{{\ddot{c}_i}^2}$, $\overline{r_i}$, $\overline{{r_i}^2}$, $\overline{d_i}$, $\overline{{d_i}^2}$, for all $1 \leq i \leq k$, ξ, α, and P^*, find the optimal ES configuration scheme $m_1, m_2, \ldots, m_k, f_1, f_2, \ldots, f_k$, such that the energy consumption of the MEC platform is minimized, namely,

$$minimize \;\; J = \sum_{i=1}^{k} \left(m_i(\rho_i \xi {f_i}^{\alpha} + P_i^*)\right),$$

subject to the constraints $\rho_i < 1$, $m_i \leq m_{\max}$, $f_i \leq f_{\max}$, for all $1 \leq i \leq k$, and $\overline{T} = \tilde{T}$.

In the first stage, we discuss for a given placement scheme, how to obtain the optimal configuration scheme. We use the Lagrange multiplier method to solve this multi-variable optimization problem. We construct a Lagrange function, namely,

$$\nabla J\big(m_1,\ldots,m_k,f_1,\ldots,f_k\big)=\phi\nabla\overline{T}(m_1,\ldots,m_k,f_1,\ldots,f_k),$$

where ϕ is a Lagrange multiplier. Then, we have $2k$ nonlinear equations, namely,

$$\begin{cases}\frac{\partial J\left(m_1,\ldots,m_k,f_1,\ldots,f_k\right)}{\partial m_i}=\phi\frac{\partial\overline{T}\left(m_1,\ldots,m_k,f_1,\ldots,f_k\right)}{\partial m_i},\\ \frac{\partial J\left(m_1,\ldots,m_k,f_1,\ldots,f_k\right)}{\partial f_i}=\phi\frac{\partial\overline{T}\left(m_1,\ldots,m_k,f_1,\ldots,f_k\right)}{\partial f_i},\end{cases}\tag{29}$$

for all $1\le i\le k$.

Notice that Equation (18) contains the factorial of m_i, and thus its partial derivative cannot be obtained directly. By using the Stirling's approximation of $m_i!$ [35], namely,

$$m_i!\approx\sqrt{2\pi m_i}\Big(\frac{m_i}{e}\Big)^{m_i},$$

and the following closed-form approximation

$$\sum_{l=0}^{m_i-1}\frac{(m_i\rho_i)^l}{l!}\approx e^{m_i\rho_i},$$

we can obtain a closed-form approximation of $\overline{T}$, that is,

$$\overline{T}=\sum_{i=1}^{k}\frac{\hat{\lambda}_i}{\lambda}\left(\overline{t_i}+\frac{\hat{\lambda}_i\overline{t_i^2}}{2}G_i\right),\tag{30}$$

where

$$G_i=\frac{1}{{m_i}^2\rho_i(1-\rho_i)\big(\sqrt{2\pi m_i}(1-\rho_i)(e^{\rho_i}/e\rho_i)^{m_i}+1\big)}.\tag{31}$$

Based on Equations (29) and (30), we can obtain

$$\phi=-\frac{2\lambda P_i^*}{m_i\rho_i{\hat{\lambda}_i}^2\overline{t_i^2}{G_i}^2\Big(N_i\Big(\frac{\rho_i+3}{2}-m_i(1-\rho_i)\ln\rho_i\Big)+1\Big)},\tag{32}$$

and

$$\phi=-\frac{\lambda\xi{f_i}^{\alpha}\Big(m_i\rho_i\alpha f_i-\hat{\lambda}_i\overline{r_i}\Big)}{\hat{\lambda}_i\overline{r_i}\left(\hat{\lambda}_iG_i\left(\left(\frac{\overline{r_i^2}}{\overline{r_i}f_i}+\frac{\overline{d_i}}{\ddot{\overline{c}}_i}\right)+M_i+\frac{\hat{\lambda}_iG_i\overline{t_i^2}m_i}{2}Q_i\right)+1\right)},\tag{33}$$

for all $1\le i\le k$, where

$$N_i=\sqrt{2\pi m_i}(1-\rho_i)\left(\frac{e^{\rho_i}}{e\rho_i}\right)^{m_i},\tag{34}$$

$$M_i=\frac{\sum_{l_v\in ne(i)}\lambda_v}{\hat{\lambda}_i}\cdot\frac{\overline{d_i}}{\ddot{\overline{c}}_i},\tag{35}$$

and

$$Q_i=N_i\Big(3\rho_i-1+m_i(1-\rho_i)^2\Big)+2\rho_i-1.\tag{36}$$

(For clarity of presentation, the derivation of Equations (32) and (33) is given in Section 2 of the supplementary file, available online.)

Let us rewrite $\overline{T}=\tilde{T}$ as

$$\sum_{i=1}^{k}\frac{\hat{\lambda}_i}{\lambda}\left(\overline{t_i}+\frac{\hat{\lambda}_i\overline{t_i^2}}{2}G_i\right)-\tilde{T}=0.\tag{37}$$

Our target is to solve $m_1,m_2,\ldots,m_k,f_1,f_2,\ldots,f_k$ based on Equations (32), (33), and (37). Solving these $2k+1$ sophisticated nonlinear equations simultaneously needs special insight, since there is absolutely no closed-form solution. The details will be discussed in Section 5.2.1.

5.1.2 Stage II

Facility location problems, including site selection of ESs, are usually regarded as NP-hard combinatorial optimization problems [36], [37], [38]. Since the cost-efficient ES configuration and placement problem is more complicated than the traditional facility location problem, it is very likely to be NP-hard.

So far, several heuristic algorithms have been proposed for solving NP-hard combinatorial optimization problems, e.g., Differential Evolution (DE), Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). Since GA-based method has a better performance in discrete problems [23], [39], in the second stage, we further design an GA-based method to solve the cost-efficient ES configuration and placement problem which is defined in Section 4. It should be noted that there may be numerous deployed BSs in a city, which means that there are more potential deployment locations for ESs and the solution space is large. In order to converge faster and avoid falling into local optimal solutions, we design a population initialization strategy based on random walk algorithm. The details will be discussed in Section 5.2.2.

5.2 Algorithms

In this section, we will implement a series of algorithms to solve the cost-efficient ES configuration and placement problem.

5.2.1 Find Optimal Configuration Scheme for a Given Placement Scheme

As mentioned in Section 5.1.1, it is impossible to obtain a closed-form solution with regard to m_i and f_i for Equations (32), (33), and (37). Thus, we propose a numerical solution based on the bisection algorithm.

Algorithm 1. *Find_lb$_\phi$*

Input: $\lambda_{h(i)}$, $\sum_{l_v\in ne(i)}\lambda_v$, $\overline{c_i}$, $\overline{c_i^2}$, $\dot{\overline{c}}_i$, $\dot{\overline{c}}_i^2$, $\ddot{\overline{c}}_i$, $\ddot{\overline{c}}_i^2$, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, for all $1\le i\le k$, α, ξ, P^*, $m_{\max}$, $f_{\max}$.
Output: lb_ϕ.
1: $lb_\phi\leftarrow$ a very big negative number; //Initialize lb_ϕ
2: **for** i in **range**(k):
3: $p_1\leftarrow\dot{\phi}(m_{\max},f_{\max})$; //Calculated by using Eq. (32)
4: $p_2\leftarrow\ddot{\phi}(m_{\max},f_{\max})$; //Calculated by using Eq. (33)
5: $lb_\phi\leftarrow p_1>p_2\,?\,p_1:p_2$;
6: **end for**
7: **return** lb_ϕ.

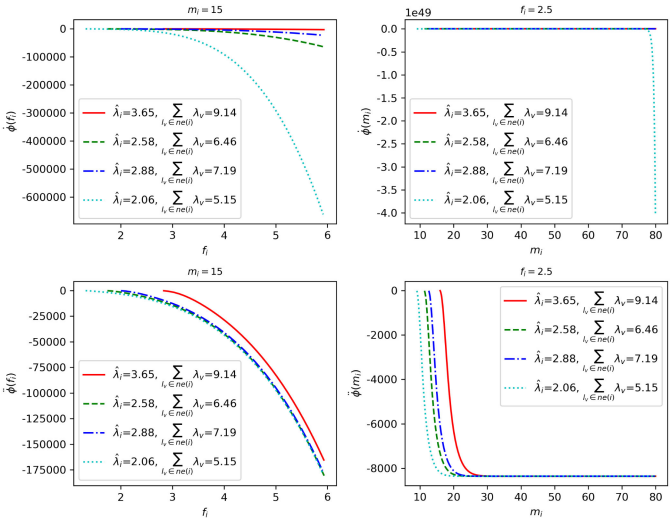

Fig. 2. Several examples of $\dot{\phi}(f_i)$, $\ddot{\phi}(f_i)$, $\dot{\phi}(m_i)$ and $\ddot{\phi}(m_i)$.

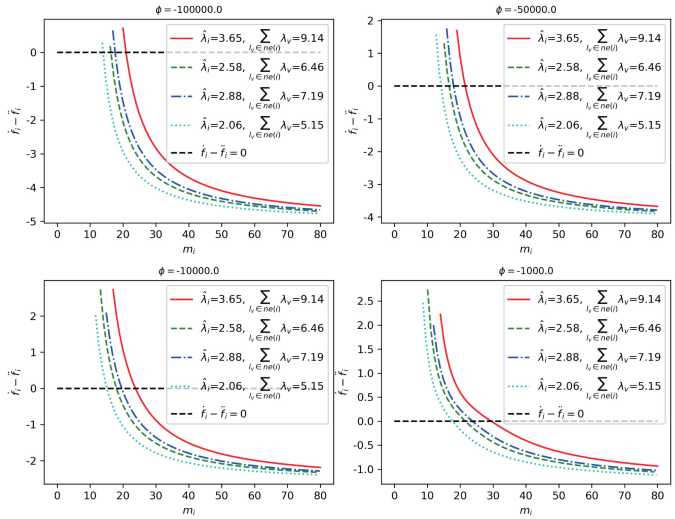

Fig. 3. Changing trend of $\dot{f}_i - \ddot{f}_i$.

A Motivating Example. This example is help for understanding our algorithms. Assume that we need to deploy 10 ESs near the selected BSs according to the given placement scheme. The parameters are set as follows: The task arrival rates $\lambda_{h(1)}, \lambda_{h(2)}, \cdots, \lambda_{h(10)}$ are random numbers that followed a normal distribution with mean 3.0 and standard deviation 1.5; $\sum_{l_v \in ne(i)} \lambda_v = 2.5\lambda_{h(i)}$, for all $1 \leq i \leq 10$; $\alpha = 3.0$; $\xi = 1.5$; $P^* = 2.0$; $m_{\max} = 80$; $f_{\max} = 6.0$. We assume that the network environment is homogeneous, that is, $\underline{\dot{c}_i} = 6.0$, $\underline{\ddot{c}_i} = 75.0$, $\overline{r_i} = 2.0$, $\overline{d_i} = 2.5$, for all $1 \leq i \leq k$, and $\underline{\dot{c}_i}^2 = 1.3\underline{\dot{c}_i}^{-2}$, $\underline{\ddot{c}_i}^2 = 1.3\underline{\ddot{c}_i}^{-2}$, $\overline{r_i}^2 = 1.3\overline{r_i}^2$, and $\overline{d_i}^2 = 1.5\overline{d_i}^2$.

Algorithm 2. *Find_f_i*

Input: ϕ, m_i, $\lambda_{h(i)}$, $\sum_{l_v \in ne(i)} \lambda_v$, $\underline{\dot{c}_i}$, $\underline{\dot{c}_i}^2$, $\underline{\ddot{c}_i}$, $\underline{\ddot{c}_i}^2$, $\overline{r_i}$, $\overline{r_i}^2$, $\overline{d_i}$, $\overline{d_i}^2$, α, ξ, P^*, $f_{\max}$.
Output: $\dot{f}_i$ or $\ddot{f}_i$ or -1 or -2.

1: Calculate lb_{f_i} by using Eq. (38);
2: $f_{\min} \leftarrow lb_{f_i}$; $ub_{f_i} \leftarrow f_{\max}$;
3: **while** True
4: $f_{mid} \leftarrow (lb_{f_i} + ub_{f_i})/2$;
5: **if** $|f_{mid} - f_{\max}| < \varepsilon$
6: **return** -1. //The search exceeds the boundary
7: **end if**
8: **if** $|f_{mid} - f_{\min}| < \varepsilon$
9: **return** -2. //The search exceeds the boundary
10: **end if**
11: $\phi_{cal} \leftarrow \dot{\phi}(f_{mid})$ or $\ddot{\phi}(f_{mid})$;
12: **if** $|\phi_{cal} - \phi| < \varepsilon$
13: **return** f_{mid}. //Return $\dot{f}_i$ or $\ddot{f}_i$
14: **end if**
15: **if** $\phi_{cal} > \phi$
16: $lb_{f_i} \leftarrow f_{mid}$;
17: **else**
18: $ub_{f_i} \leftarrow f_{mid}$;
19: **end if**
20: **end while**

If the value of m_i is fixed, then Equations (32) and (33) can be regarded as functions of f_i, that is, $\dot{\phi}(f_i)$ and $\ddot{\phi}(f_i)$, respectively. Since $\rho_i < 1$ and $f_i \leq f_{\max}$, we can obtain that the lower bound of f_i is

$$lb_{f_i} = \frac{\hat{\lambda}_i \overline{r_i}}{m_i - \hat{\lambda}_i \frac{\overline{d_i}}{\underline{\dot{c}_i}} - \sum_{l_v \in ne(i)} \lambda_v \frac{\overline{d_i}}{\underline{\dot{c}_i}}}, \tag{38}$$

and the upper bound of f_i is $ub_{f_i} = f_{\max}$. Similarly, for a given f_i, Equations (32) and (33) can be regarded as functions of m_i, namely, $\dot{\phi}(m_i)$ and $\ddot{\phi}(m_i)$. The lower bound of m_i can be calculated as

$$lb_{m_i} = \hat{\lambda}_i \left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\underline{\dot{c}_i}} \right) + \sum_{l_v \in ne(i)} \lambda_v \frac{\overline{d_i}}{\underline{\dot{c}_i}}, \tag{39}$$

and the upper bound of m_i is $ub_{m_i} = m_{\max}$. From our observations, we find that $\dot{\phi}(f_i)$, $\ddot{\phi}(f_i)$, $\dot{\phi}(m_i)$ and $\ddot{\phi}(m_i)$ are all decreasing functions in the domain $[lb_{f_i}, ub_{f_i}]$ or $[lb_{m_i}, ub_{m_i}]$, and $\phi < 0$. Fig. 2 shows several examples of $\dot{\phi}(f_i)$, $\ddot{\phi}(f_i)$, $\dot{\phi}(m_i)$ and $\ddot{\phi}(m_i)$. Thus, we can use Algorithm 1 to find the lower bound of ϕ, namely, lb_ϕ. Since $\phi < 0$, we set the upper bound of ϕ as $ub_\phi = -0.1$ in this paper. Given the values of ϕ and m_i, we can obtain two values of f_i (namely, $\dot{f}_i$ and $\ddot{f}_i$, respectively) through Algorithm 2, such that $\dot{\phi}(\dot{f}_i) = \phi$ and $\ddot{\phi}(\ddot{f}_i) = \phi$. (We set $\varepsilon = 10^{-7}$ in this paper.) The value of ϕ is determined by both m_i and f_i. If the value of m_i is inappropriate, the search for f_i may exceed the boundary. Thus, we judge the situation in Algorithm 2, and return the specific values -1 or -2 to help us adjust the value of m_i (lines 5-10). Due to the different changing trends of functions $\dot{\phi}(f_i)$ and $\ddot{\phi}(f_i)$, the values of $\dot{f}_i$ and $\ddot{f}_i$ may be different. Through our further observation, we find that the value of $\dot{f}_i - \ddot{f}_i$ will gradually decrease with the increase of m_i. Fig. 3 shows the changing trend of $\dot{f}_i - \ddot{f}_i$. Therefore, given the value of ϕ (i.e., $\phi \in [lb_\phi, ub_\phi]$), we can obtain the values of m_i and f_i through Algorithm 3, such that Equations (32) and (33) hold. Since the value of ϕ determines the values of m_i and f_i, for all $1 \leq i \leq k$, the value of ϕ indirectly determines whether Equation (37) holds. We find that the average response time of all offloadable tasks will gradually increase with the increase of ϕ (Shown in Fig. 4). Therefore, we can use Algorithm 4 to find the value of ϕ and the values of m_i and f_i, for all $1 \leq i \leq k$, such that Equations (32), (33), and

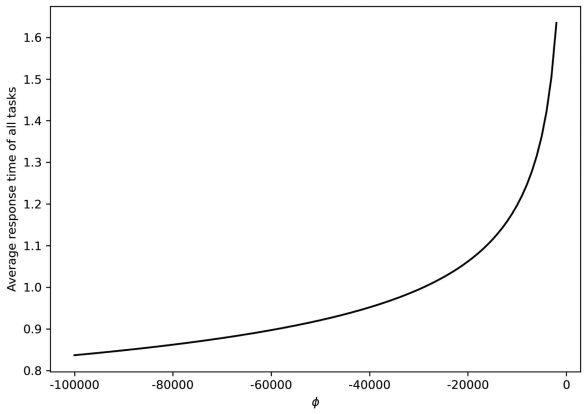

Fig. 4. Changing trend of $\overline{T}$.

TABLE 1
Optimal Configuration Scheme of the Example ($\tilde{T} = 1.0$)

i	$\lambda_{h(i)}$	$\sum_{l_v \in ne(i)} \lambda_v$	m_i	f_i
1	5.682943	14.207357	31	3.578859
2	3.654765	9.136912	22	3.579390
3	3.144746	7.861866	19	3.579600
4	0.204761	0.511902	3	3.588699
5	2.583918	6.459794	17	3.579901
6	2.467862	6.169654	16	3.579976
7	2.875888	7.189719	18	3.579733
8	2.059499	5.148747	14	3.580289
9	2.934273	7.335682	18	3.579703
10	2.284173	5.710432	15	3.580106
$\phi = -29040.980645$, $\overline{T} = 1.000359$, $J = 7057.606221$				

(37) hold. However, our algorithms regards the server sizes $m_1, m_2, \ldots, m_k$ as a series of continuous values, but the number of processors can only be a positive integer. Thus, we need to further round the server sizes to the nearest integers.

Algorithm 3. *Find_m_i_f_i*

Input: ϕ, $\lambda_{h(i)}$, $\sum_{l_v \in ne(i)} \lambda_v$, $\overline{\dot{c}_i}$, $\overline{\dot{c}_i{}^2}$, $\overline{\ddot{c}_i}$, $\overline{\ddot{c}_i{}^2}$, $\overline{r_i}$, $\overline{r_i{}^2}$, $\overline{d_i}$, $\overline{d_i{}^2}$, α, ξ, P^*, $m_{\max}$, $f_{\max}$.
Output: m_i and f_i.
1: $lb_{m_i} \leftarrow \hat{\lambda}_i \left(\frac{\overline{r_i}}{f_{\max}} + \frac{\overline{d_i}}{\overline{\dot{c}_i}} \right) + \sum_{l_v \in ne(i)} \lambda_v \frac{\overline{d_i}}{\overline{\dot{c}_i}}$;
2: $ub_{m_i} \leftarrow m_{\max}$;
3: **while** True
4: $m_{mid} \leftarrow (lb_{m_i} + ub_{m_i})/2$;
5: Call Algorithm 2 (with m_{mid}) to obtain $\dot{f}_i, \ddot{f}_i$;
6: **if** $\dot{f}_i == -1$ or $\ddot{f}_i == -1$
7: $lb_{m_i} \leftarrow m_{mid}$;
8: **else if** $\dot{f}_i == -2$ or $\ddot{f}_i == -2$
9: $ub_{m_i} \leftarrow m_{mid}$;
10: **else**
11: $df \leftarrow \dot{f}_i - \ddot{f}_i$;
12: **if** $|df| < \varepsilon$
13: **return** $m_{mid}, (\dot{f}_i + \ddot{f}_i)/2$. //Return m_i and f_i
14: **else if** $df > 0$
15: $lb_{m_i} \leftarrow m_{mid}$;
16: **else**
17: $ub_{m_i} \leftarrow m_{mid}$;
18: **end if**
19: **end if**
20: **end while**

TABLE 2
Optimal Configuration Scheme of the Example ($\tilde{T} = 0.8$)

i	$\lambda_{h(i)}$	$\sum_{l_v \in ne(i)} \lambda_v$	m_i	f_i
1	5.682943	14.207357	28	5.564758
2	3.654765	9.136912	20	5.564972
3	3.144746	7.861866	18	5.565057
4	0.204761	0.511902	3	5.568792
5	2.583918	6.459794	16	5.565178
6	2.467862	6.169654	15	5.565208
7	2.875888	7.189719	17	5.565110
8	2.059499	5.148747	13	5.565335
9	2.934273	7.335682	17	5.565098
10	2.284173	5.710432	14	5.565261
$\phi = -142972.900525$, $\overline{T} = 0.800129$, $J = 20509.421690$				

Tables 1 and 2 respectively illustrate the optimal configuration scheme of the example under the performance constraints $\tilde{T} = 1.0$ and $\tilde{T} = 0.8$. The implication of these results is that the processors configured on ESs can choose the same type, since the difference between their optimal speeds is very small.

5.2.2 Find Sub-Optimal Placement Scheme and Optimal Configuration Scheme for ESs

Based on the algorithms proposed in Stage I, we further design a heuristic algorithm to solve the cost-efficient ES configuration and placement problem.

Algorithm 4. *Find_ϕ_m_i_f_i*

Input: $\lambda_{h(i)}$, $\sum_{l_v \in ne(i)} \lambda_v$, $\overline{\dot{c}_i}$, $\overline{\dot{c}_i{}^2}$, $\overline{\ddot{c}_i}$, $\overline{\ddot{c}_i{}^2}$, $\overline{r_i}$, $\overline{r_i{}^2}$, $\overline{d_i}$, $\overline{d_i{}^2}$, for all $1 \le i \le k$, α, ξ, P^*, $m_{\max}$, $f_{\max}$, $\tilde{T}$.
Output: m_i, f_i, for all $1 \le i \le k$, and ϕ.
1: Call Algorithm 1 to obtain lb_ϕ;
2: $ub_\phi \leftarrow -0.1$;
3: **while** True
4: $\phi_{mid} \leftarrow (lb_\phi + ub_\phi)/2$;
5: **for** i in **range**(k):
6: Call Algorithm 3 (with ϕ_{mid}) to obtain m_i, f_i;
7: **end for**
8: Calculate $\overline{T}$ by using Eq. (30);
9: **if** $\left|\overline{T} - \tilde{T}\right| < \varepsilon$
10: **return** m_i, f_i, for all $1 \le i \le k$, and ϕ_{mid}.
11: **else if** $\overline{T} > \tilde{T}$
12: $ub_\phi \leftarrow \phi_{mid}$;
13: **else**
14: $lb_\phi \leftarrow \phi_{mid}$;
15: **end if**
16: **end while**

Encoding Scheme. Recall that $u_1, u_2, \ldots, u_n$, $u_j \in \{0, 1\}$ denotes a placement scheme of ESs. Therefore, it can be naturally expressed as a chromosome, e.g., if $n = 10$ and we will deploy three ESs at the locations of the 1th, 6th, and 7th BSs, then the chromosome can be encoded as 1000011000. Obviously, there are 2^n different encoding schemes corresponding to 2^n different placement schemes. However, these placement schemes are not all reasonable. There are two unreasonable situations caused by the insufficient number of ESs, as described below.

- The workload of ES exceed its maximum computation capacity. For a placement scheme, if $\exists s_i \in S$ and

$$\rho_i^{lb} = \frac{\hat{\lambda}_i}{m_{\max}} \left(\frac{\overline{r_i}}{f_{\max}} + \frac{\overline{d_i}}{\overline{\dot{c}_i}} \right) + \frac{\sum_{l_v \in ne(i)} \lambda_v}{m_{\max}} \frac{\overline{d_i}}{\overline{\dot{c}_i}} > 1, \tag{40}$$

  then the placement scheme is unreasonable.
- The minimum average response time cannot meet the given performance constraint. For a placement scheme, if $\rho_i^{lb} < 1$, for all $1 \le i \le k$, we can obtain lb_ϕ through Algorithm 1. Since the average response time will gradually decrease with the decrease of ϕ, we can get configuration scheme with lb_ϕ through Algorithm 3, and then calculate the minimum average response time $\overline{T}_{lb}$. If $\overline{T}_{lb} > \tilde{T}$, the placement scheme is unreasonable.

In the first stage, we have an implicit assumption that the given ES placement scheme is reasonable. In this stage, we need to determine the rationality of the placement schemes (i.e., Algorithm 5).

Population Initialization. As mentioned in Section 5.1.2, we propose a population initialization strategy based on random walk algorithm to accelerate convergence and avoid falling into local optimal solutions. Let p denote the population size, and Algorithm 6 describes the main steps to obtain the initial population. The main idea of Algorithm 6 is to walk randomly among the BSs, and deploy an ES at a different BS each time until the placement scheme is reasonable.

Algorithm 5. *Determine_Rationality*

Input: u_j, l_j, λ_j, for all $1 \le j \le n$, $\overline{\dot{c}_i}$, $\overline{\dot{c}_i^2}$, $\overline{\ddot{c}_i}$, $\overline{\ddot{c}_i^2}$, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, for all $1 \le i \le k$, α, ξ, P^*, $m_{\max}$, $f_{\max}$, $\tilde{T}$.
Output: *True* or *False*.

```
Assign BSs to the nearest ES, then calculate λ_{h(i)} and Σ_{l_v∈ne(i)} λ_v, for all 1 ≤ i ≤ k.
for i in range(k):
    if ρ_i^{lb} > 1
        return False.
    end if
end for
Call Algorithm 1 to obtain lb_φ;
for i in range(k):
    Call Algorithm 3 (with lb_φ) to obtain m_i, f_i;
end for
Calculate T̄_{lb} by using Eq. (30);
if T̄_{lb} > T̃
    return False.
end if
return True.
```

Crossover and Mutation. In order to generate new offspring, we divide the individuals in the population into two groups on average and perform crossover operations, that is, exchange a random segment of the two chromosomes in each row of the two groups. Since each offspring carries the genes of both parents, after many iterations, the chromosomes in the population may become similar to each other. To maintain genetic diversity and reduce the risk of the algorithm falling into a local optimal solution, we also need to perform mutation operations, that is, change the value of multiple random bits in each chromosome according to mutation factor (i.e., the number of changed bits equal to mutation factor).

Algorithm 6. *Initialize_Population*

Input: l_j, λ_j, for all $1 \le j \le n$, $\overline{\dot{c}_i}$, $\overline{\dot{c}_i^2}$, $\overline{\ddot{c}_i}$, $\overline{\ddot{c}_i^2}$, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, for all $1 \le i \le k$, α, ξ, P^*, $m_{\max}$, $f_{\max}$, $\tilde{T}$, p.
Output: Initial population.

```
pop ← [ ];
while p > 0
    u_j ← 0, for all 1 ≤ j ≤ n; // Initialize scheme
  do
    Randomly select an BS b_j from B;
    if u_j == 0
      u_j ← 1;
      Call Algorithm 5 to obtain reasonable;
    else
      continue;
    end if
  while reasonable == False
  Express the placement scheme (u_1, u_2, ··· , u_n) as a chromosome chorm;
    pop.append(chorm);
    p ← p − 1;
end while
return pop.
```

Fitness Function. In order to measure the quality of different chromosomes, we define the fitness function as

$$fit(u_1, u_2, \ldots, u_n) = \begin{cases} \left|\rho_i^{lb} > 1\right| \cdot M, \ 1 \le i \le k, & \text{(41a)} \\ \left|\overline{T}_{lb} - \tilde{T}\right| \cdot M, & \text{(41b)} \\ C, & \text{(41c)}. \end{cases}$$

where M denotes the penalty factor. (We set $M = 10^{20}$ in this paper.) After crossover and mutation operations, the newly generated placement schemes may be unreasonable. For a placement scheme, if $\exists s_i \in S$ and $\rho_i^{lb} > 1$, we use Equation (41a) to calculate the fitness value, where $\left|\rho_i^{lb} > 1\right|$ denotes the number of ESs whose workload exceed its maximum computation capacity; if $\overline{T}_{lb} > \tilde{T}$, we use Equation (41b) to calculate the fitness value; if the placement scheme is reasonable, we use Algorithm 4 to obtain the optimal configuration scheme, then calculate the OPEX as the fitness value.

Selection. In order to screen better individuals and generate new population, we first merge the parent chromosomes and offspring chromosomes into one population, and then use roulette wheel method to select p chromosomes to form a new population according to their fitness values. After the population information is updated, we enter the next iteration.

Algorithm 7 describes the steps that get the final schemes. The parameter p denotes the population size, max_iter denotes the maximum iteration number, pop is a list which stores the population chromosomes, and $best\_chromosome$ stores the chromosome with the smallest fitness value (i.e., the minimum OPEX).

6 Experimental Evaluation

In this section, we conduct the experiment based on a real base station dataset collected in Shanghai. Experimental

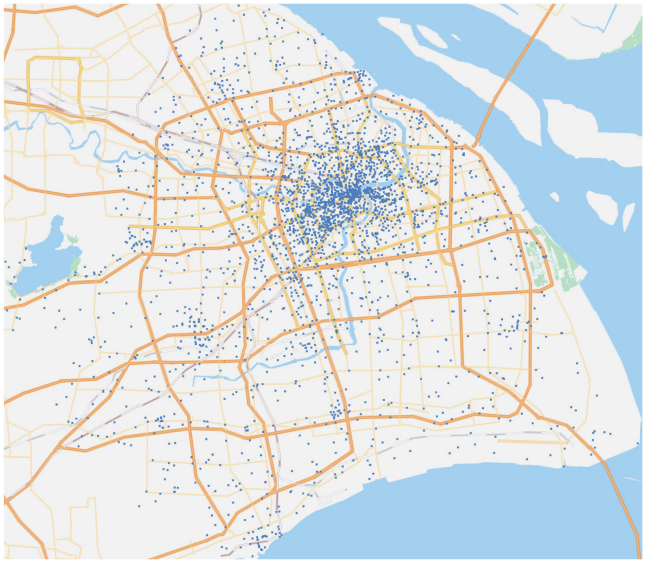

Fig. 5. Distribution of BSs in the dataset.

results show that the algorithms proposed in this paper are more effective than several other representative algorithms.

Algorithm 7. *Obtain_Schemes*

Input: l_j, λ_j, c_j, for all $1 \leq j \leq n$, $\overline{\dot{c}_i}$, $\overline{\dot{c}_i^2}$, $\overline{\ddot{c}_i}$, $\overline{\ddot{c}_i^2}$, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, for all $1 \leq i \leq k$, α, ξ, P^*, $m_{\max}$, $f_{\max}$, κ, C_e, p, max_iter, $\tilde{T}$.
Output: Configuration and placement schemes of ESs

```
//Initialize the parameters
best_chromosome ← null;
min_fit_val ← a very large value;
Call Algorithm 6 to obtain the initial population and store it in the list pop;
while max_iter ≥ 1
    Perform crossover and mutation operations;
    Add the new offspring chromosomes to the list pop;
    fit_vals ← [ ];
    for chromosome in pop
        Decode chromosome, assign BSs to the nearest ES;
        Calculate fit_val by using Eq. (41);
        if min_fit_val > fit_val
            min_fit_val ← fit_val;
            best_chromosome ← chromosome;
    end if
            fit_vals.append(fit_val);
    end for
        pop ← [ ];
     According to fit_vals, use the roulette wheel method to select p chromosomes and store them in the list pop;
        max_iter ← max_iter − 1;
end while
Decode best_chromosome, assign BSs to the nearest ES;
Obtain optimal configuration scheme by Algorithm 4;
return Configuration and placement schemes of ESs.
```

6.1 Experiment Setup

In our experiment, we use a real-world dataset provided by Shanghai Telecom [40], which contains internet information for mobile users who access 3233 BSs in Shanghai. According to our analysis, the information of 3008 BSs in the dataset are valid, while the information for the remaining BSs are invalid due to null values or incorrect longitude and latitude values. Fig. 5 illustrates the distribution of valid BSs (which marked with blue dots). The dataset also contains the detailed start time and end time of each mobile user accessing the Internet through BSs. We use the average number of user access per half hour to estimate the task arrival rate of each BS (i.e., λ_j). For each candidate placement scheme, we assign each BS to the nearest ES to calculate the values of $\lambda_{h(i)}$ and $\sum_{l_v \in ne(i)} \lambda_v$, for all $1 \leq i \leq k$. In order to evaluate the site rentals, we implemented a crawler program using Python and obtained 23291 house rental information of various regions in Shanghai form LianJia website [41]. Then, we used the coordinate pickup system provided by Baidu Map [42] to obtain the locations of these rental houses. For each BS, we use the average annual rental per square meter of the nearest 10 rental houses multiplied by 20 (i.e., assume that the average computer room area is 20 square meters) to estimate the site rental of the location. Table 3 illustrates the task arrival rates of several base stations in the dataset and the site rentals at their locations. Other parameters of this experiment are set as follows: $\overline{\dot{c}_i} = 6.0$, $\overline{\ddot{c}_i} = 75.0$, $\overline{r_i} = 2.0$, $\overline{d_i} = 2.5$, for all $1 \leq i \leq k$; $\overline{\dot{c}_i^2} = 1.3\overline{\dot{c}_i}^2$; $\overline{\ddot{c}_i^2} = 1.3\overline{\ddot{c}_i}^2$; $\overline{r_i^2} = 1.3\overline{r_i}^2$; $\overline{d_i^2} = 1.5\overline{d_i}^2$; $\alpha = 3.0$; $\xi = 1.5$; $P^* = 2.0$; $m_{\max} = 80$; $f_{\max} = 6.0$; $\tilde{T} = 0.8$; $\kappa = 3$; $C_e = 0.917/1000/3600$; $p = 50$; $max\_iter = 150$ (According to our experimental results, the algorithm converges after about 120 iterations when $n = 3008$).

TABLE 3
Task Arrival Rates and Site Rentals of Several Base Stations

BS Id	Task Arrival Rate (tasks/second)	Site Rental (CNY/year)
27	1.490306	6482.40
115	1.507408	5568.45
640	1.645290	15059.16
797	2.875910	22490.57
878	9.346994	20155.51
890	3.892339	21654.76
1136	2.569926	23660.65
1714	1.669642	44450.11
2034	2.516697	15669.45
2942	1.785882	14271.58

6.2 Comparative Algorithms

Notice that all the following comparative algorithms obtain the optimal configuration scheme through the algorithms we proposed in Stage I. They differ from each other only when selecting ES placement scheme. Thus, the comparison experiment mainly shows the effectiveness of Stage II.

6.2.1 K-Means++

Centroid-based clustering algorithms usually generate clusters of similar sizes. We can automatically cluster n BSs into k groups based on the spherical distance (i.e., Equation (3)), and deploy k ESs at the BSs nearest to the k centroids to obtain a placement scheme. Since K-means algorithm is sensitive to the initialization of centers and the clustering result is not stable, for $k = 1$ to n, we use K-means++ algorithm to cluster n BSs into k groups, and use Algorithm 5 to determine the rationality of the corresponding placement scheme

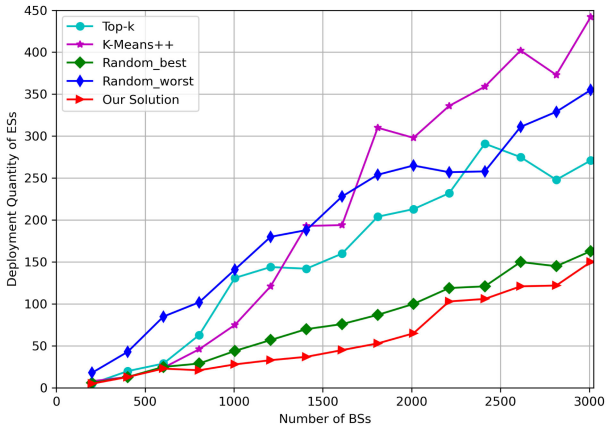

Fig. 6. Comparison of ES Deployment Quantity.

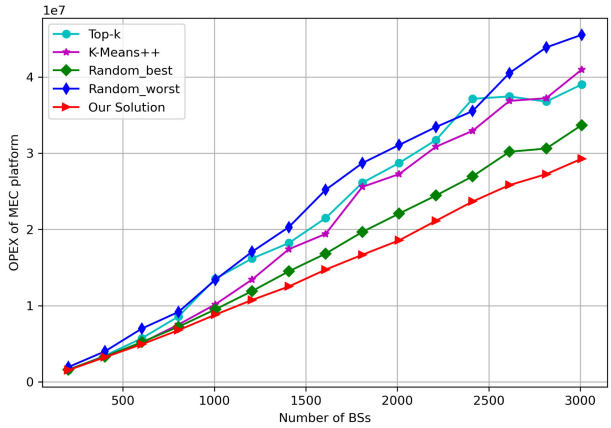

Fig. 7. Comparison of OPEX.

until it returns True. This approach is used to obtain a reasonable placement scheme with evenly distributed ESs. Then, we use Algorithm 4 to obtain the optimal configuration scheme.

6.2.2 Top-k

Deploying ESs at the BSs with heavy computation demands can reduce the transmission latency caused by task migration. For $k = 1$ to n, we select Top-k busiest BSs that have more task arrival rates than others to deploy k ESs, and use Algorithm 5 to determine the rationality of the corresponding placement scheme until it returns True. Then, we use Algorithm 4 to obtain the optimal configuration scheme.

6.2.3 Random

For $k = 1$ to n, we randomly select k BSs to deploy k ESs, and use Algorithm 5 to determine the rationality of the corresponding placement scheme until it returns True. This approach is used to obtain a reasonable placement scheme with randomness. Then, we use Algorithm 4 to obtain the optimal configuration scheme.

6.3 Comparison Analysis

In this section, we evaluate the effectiveness of different methods based on the dataset and parameter settings described in Section 6.1. To do this, we increase the number of BSs n from 200 to 3008, and adopt different methods to obtain the corresponding ES configuration and placement schemes under the premise of the system performance is close to the given performance constraint. Figs. 6 and 7 respectively illustrate the curve of ES deployment quantity and OPEX of different schemes with the increase of BSs quantity. It should be noted that we use Random method to conduct 100 experiments for each value of n, and illustrate the best and worst experimental results among them.

According to Figs. 6 and 7, we can see that for all methods, the number of ESs that need to be deployed and the platform OPEX will increase with the number of BSs under consideration. With respect to the ES deployment quantity, the ranking is K-Means++ $>$ Top-k $>$ our solution. In terms of OPEX, the ranking is Top-k $>$ K-Means++ $>$ our solution. In 100 experiments performed by Random method, the ES deployment quantity and OPEX obtained by Random method is less than that obtained by K-Means++ and Top-k methods in the best case, but may not be better than K-Means++ and Top-k methods in the worst case. Since our solution has the least ES deployment quantity and OPEX, our strategy is more effective than others.

In order to understand the experimental results more intuitively, we visualize the final placement schemes (i.e., consider all BSs, $n = 3008$) obtained by different methods, as shown in Figs. 8, 9, 10, 11, and 12. The red squares indicate the deployment locations of ESs, and the blue dots represent the locations of BSs. According to these figures, we have the following analysis results.

- As K-Means++ is a distance-based clustering algorithm, the ESs will be more evenly placed geographically, such that the number of deployed ES is the largest.
- The Top-k method selects the busiest BSs to deploy ESs. However, these BSs are often located in densely populated areas, such as shopping malls, residential areas, and transportation hubs, which means that the site rentals in these locations are higher than others. This factor leads to the higher OPEX.

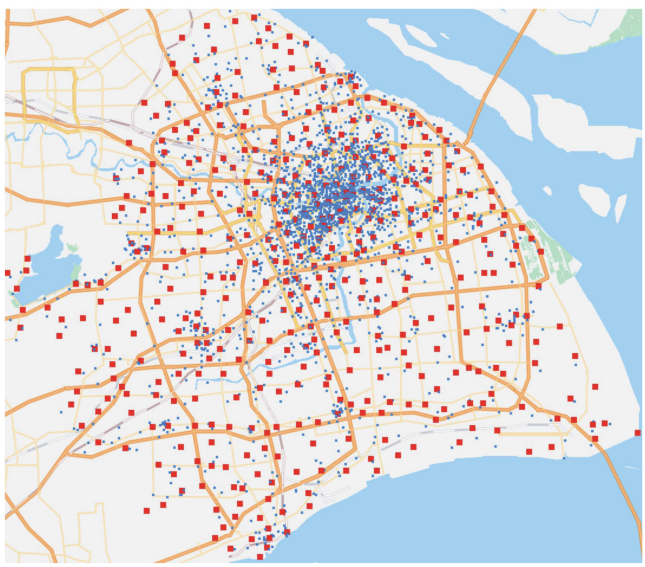

Fig. 8. Distribution map with 442 ESs by K-Means++.

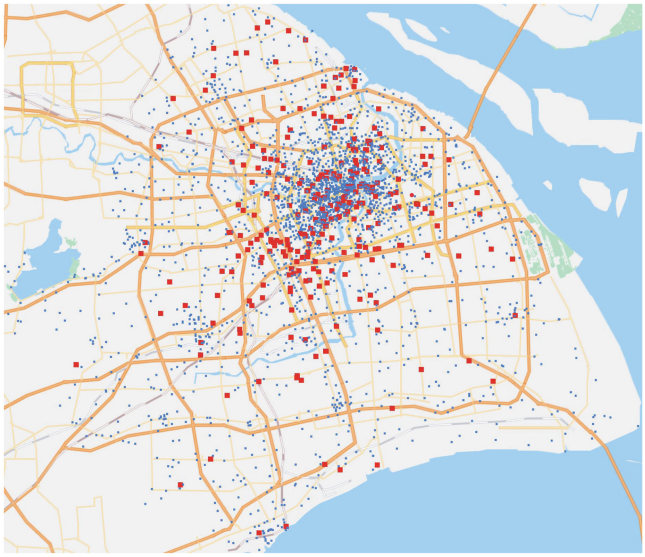

Fig. 9. Distribution map with 271 ESs by Top-k.

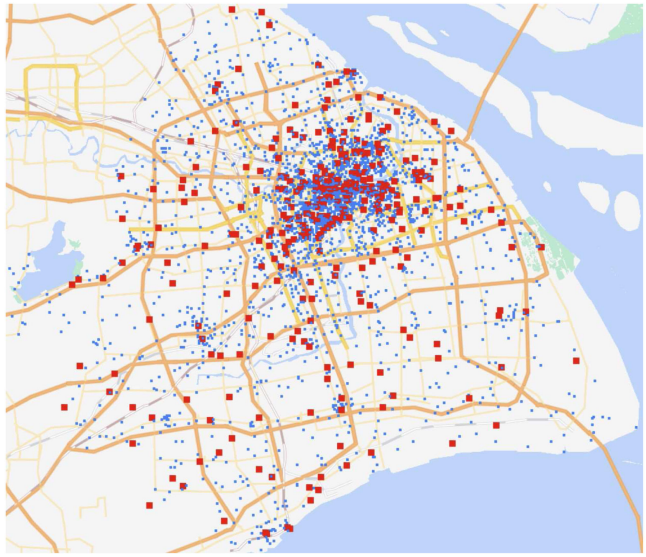

Fig. 11. Distribution map with 355 ESs by Random (the worst case in 100 experiments).

- In the 100 experiments conducted with the Random method, the solution obtained by the Random method is better than the solutions obtained by K-Means++ and Top-k in the best case, but worse than the solutions obtained by other methods in the worst case. The solutions obtained by random algorithm are different every time, such that it has diversity and is suitable as a population initialization strategy.
- The placement scheme obtained by our strategy is relatively balanced. The ESs are mainly placed in two types of areas, namely, areas with higher computation demands and areas with lower site rentals. Therefore, it can better balance the impact of site rentals and computation demands on OPEX.

Due to space limitations, the corresponding ES configuration schemes are shown in Section 3 of the supplementary file, available online.

6.4 Comparison with Optimal Solution

In this section, we construct a comparison between the solution obtained by our strategy and the optimal solution obtained by the traversal method on a small-scale problem instance, in order to evaluate the proximity of the solution we obtained to the optimal solution, and the tradeoff between the solution quality and the algorithm running time.

The main idea of the traversal method we use is to apply Algorithm 4 to obtain the optimal ES configuration scheme under all possible ES placement schemes, and then select the ES configuration and placement scheme with the smallest OPEX as the global optimal solution. First, we select 15 BSs from the Shanghai Telecom dataset as the small-scale problem instance. Second, we increase the number of BSs n from 5 to 15, and adopt our algorithms and traversal method to obtain the corresponding ES configuration and placement schemes, respectively. Third, we compare the closeness of the OPEX obtained by our strategy and the

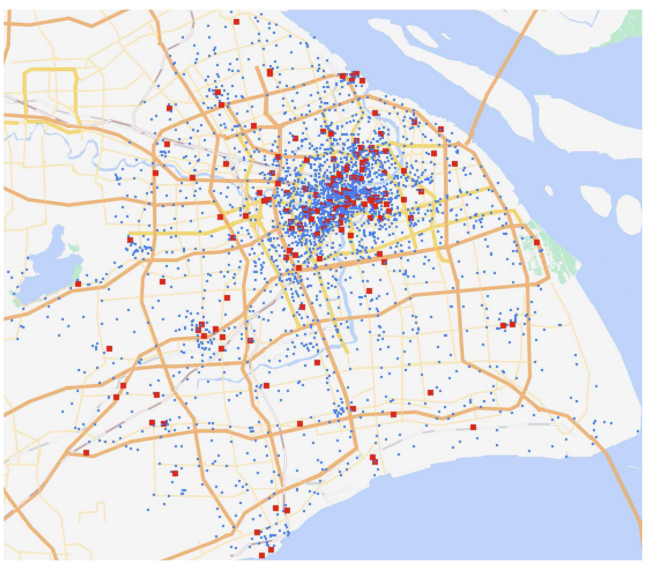

Fig. 10. Distribution map with 163 ESs by Random (the best case in 100 experiments).

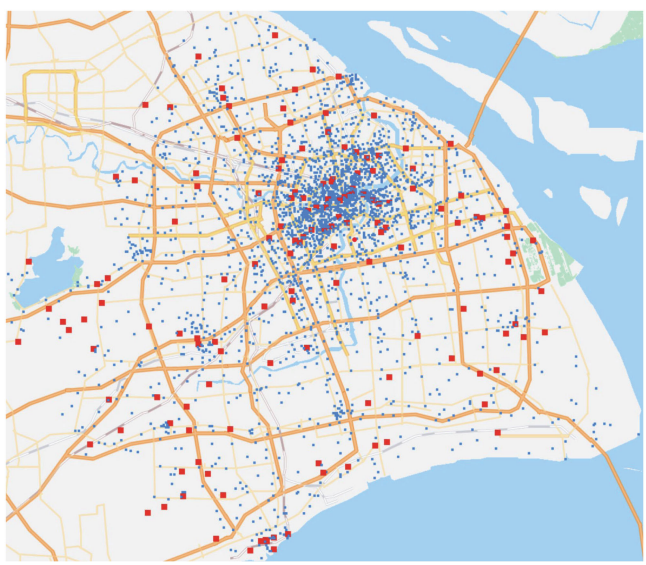

Fig. 12. Distribution map with 150 ESs by our strategy.

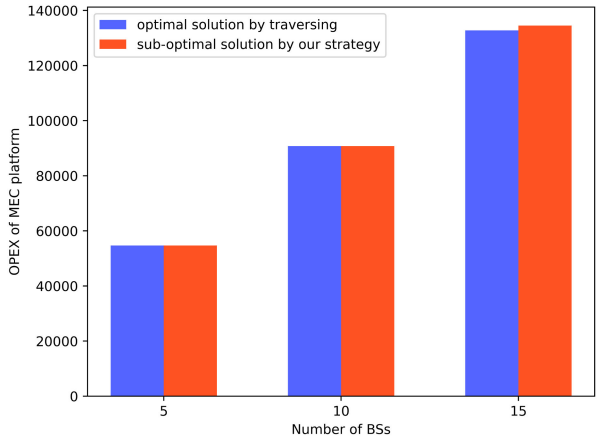

Fig. 13. Comparison of solution quality.

traversal method. It should be noted that the reason why we only selected 15 BSs as the small-scale problem instance for comparison is that we can hardly get the global optimal solution through the traversal method when the number of BSs is greater than 15 (since the running time of traversal algorithms increases exponentially).

In this comparison, we reduce the maximum iteration number (i.e., max_iter) in our strategy from 150 to 50, except that the other parameter settings are exactly the same as those in Section 6.1. We perform this comparison experiment on a computer with Intel(R) Core(TM) i7-7820X @ 3.60GHz CPU, and 64 GB RAM. Fig. 13 illustrates the comparison result.

According to Fig. 13, we can see that the sub-optimal solution we obtained on the small-scale problem instance is very close to the optimal solution. In addition, we find that the running time of our algorithms increases slowly with the expansion of problem scale, rather than exponentially increasing like the traversal algorithms. For example, when the number of BSs in the environment is 15, the running time of our algorithms to obtain the sub-optimal solution is 3203.93 seconds, but the running time of the traversal algorithms to obtain the optimal solution is 200956.585 seconds.

7 Conclusion

In this paper, we have mentioned the significance of server configuration and placement optimization in MEC. We have reviewed the existing related research and summarized the flaws of these studies. From the perspective of MNO, we have investigated the joint optimization problem of configuration and placement for ES in the MEC environment, where the main objective is to increase the cost efficiency. Taking into account the dependence of the placement scheme on the configuration scheme, we have designed a two-stage method and develop a series of algorithms to obtain the optimal deployment quantity and locations of ESs, and the optimal number and speed of processors for each ES. Our algorithms only need to be used once before system deployment, and can be reused when the environment changes significantly. The experiments are conducted based on a real base station dataset provided by Shanghai Telecom, and the results prove that our proposed algorithms is effective. The research results of this paper provide theoretical and practical insights into the establishment of MEC platforms.

Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments are very important to improve this manuscript.

References

[1] Cisco annual internet report (2018–2023) white paper. Cisco, San Jose, CA, USA, White Paper, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11–741490.html

[2] C. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," Feb., 2019. Accessed: Dec. 22, 2021. [Online]. Available: https://s3.amazonaws.com/media.mediapost.com/uploads/CiscoForecast.pdf

[3] W. Shi, G. Pallis, and Z. Xu, "Edge computing [scanning the issue]," *Proc. IEEE*, vol. 107, no. 8, pp. 1474–1481, Aug. 2019.

[4] A. Santoyo González, "Edge computing infrastructure for 5G networks: A placement optimization solution," Ph.D. dissertation, Dept. Netw. Eng. Universitat Politécnica de Catalunya, 2020. Accessed: Dec. 22, 2021. [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/328947/TASG1de2.pdf

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.

[6] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2012.

[7] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Jul.–Sep. 2021.

[8] A. Samanta, F. Esposito, and T. G. Nguyen, "Fault-tolerant mechanism for edge-based IoT networks with demand uncertainty," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16963–16971, Dec. 2021.

[9] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[10] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6164–6174, Jul. 2020.

[11] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[12] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3864–3872, Apr. 2019.

[13] A. Samanta, L. Jiao, M. Mühlhäuser, and L. Wang, "Incentivizing microservices for online resource sharing in edge clouds," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 420–430.

[14] Z. He, K. Li, K. Li, and W. Zhou, "Server configuration optimization in mobile edge computing: A cost-performance tradeoff perspective," *Softw. Pract. Experience*, vol. 51, no. 9, pp. 1868–1895, 2021.

[15] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 16, no. 2, pp. 459–474, Jun. 2019.

[16] S. Mondal, G. Das, and E. Wong, "CCOMPASSION: A hybrid cloudlet placement framework over passive optical access networks," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, 2018, pp. 216–224.

[17] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

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

[19] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2015.
[20] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct.–Dec. 2015.
[21] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," *IEEE Trans. Netw. Serv. Manage.*, vol. 15, no. 2, pp. 787–796, Jun. 2018.
[22] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput.*, 2018, pp. 66–73.
[23] X. Xu *et al.*, "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2910–2918, Apr. 2020.
[24] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven VNF placement in a MEC-NFV environment," in *Proc. IEEE Glob. Commun. Conf.*, 2018, pp. 1–7.
[25] Y. Ma, W. Liang, M. Huang, W. Xu, and S. Guo, "Virtual network function service provisioning in MEC via trading off the usages between computing and communication resources," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.3043313.
[26] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online VNF lifecycle management in an MEC-enabled 5G IoT architecture," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4183–4194, May 2020.
[27] M. Wang, B. Cheng, W. Feng, and J. Chen, "An efficient service function chain placement algorithm in a MEC-NFV environment," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.
[28] N. Kiran, X. Liu, S. Wang, and C. Yin, "VNF placement and resource allocation in SDN/NFV-enabled MEC networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Work.*, 2020, pp. 1–6.
[29] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Serv.*, 2010, pp. 49–62.
[30] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, 2013.
[31] A. Lee and P. Longton, "Queueing processes associated with airline passenger check-in," *J. Oper. Res. Soc.*, vol. 10, no. 1, pp. 56–71, 1959.
[32] L. Kleinrock, *Queueing Systems. Volume I: Theory*. New York, NY, USA: Wiley, 1975.
[33] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2019.2904680.
[34] K. Li, "Heuristic computation offloading algorithms for mobile users in fog computing," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 2, pp. 1–28, 2021.
[35] D. Romik, "Stirling's approximation for n! The ultimate short proof?," *Amer. Math. Monthly*, vol. 107, no. 6, pp. 556–557, 2000.
[36] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*, New York, NY, USA: Wiley, 1990.
[37] D. B. Shmoys, É. Tardos, and K. Aardal, "Approximation algorithms for facility location problems," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, 1997, pp. 265–274.
[38] S. Martello, M. Minoux, C. Ribeiro, and G. Laporte, *Surveys in Combinatorial Optimization*. Amsterdam, Netherlands: Elsevier, 2011.
[39] M. Akbari and M. Henteh, "Comparison of genetic algorithm (GA) and particle swarm optimization algorithm (PSO) for discrete and continuous size optimization of 2D truss structures," *J. Soft Comput. Civil Eng.*, vol. 3, no. 2, pp. 76–97, 2019.
[40] "The telecom dataset," Accessed: Dec. 22, 2021. [Online]. Available: http://sguangwang.com/TelecomDataset.html
[41] "Lianjia," Accessed: Dec. 22, 2021. [Online]. Available: https://cs.lianjia.com/
[42] "Coordinate pickup system," Accessed: Dec. 22, 2021. [Online]. Available: http://api.map.baidu.com/lbsapi/getpoint/

Zhenli He received the PhD degree in systems analysis and integration from Yunnan University, China, in 2015. He is currently a lecturer with the School of Software, Yunnan University, China. He is currently a postdoctoral researcher with Hunan University, Changsha, China. His research interests include edge computing, energy-efficient computing, heterogeneous computing, and machine learning.

Kenli Li (Senior Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. From 2004 to 2005, he was a visiting scholar with University of Illinois, Urbana-Champaign. He is currently a full professor of computer science and technology with Hunan University and the deputy director of National Supercomputing Center, Changsha. He has authored or coauthored more than 150 papers in international conferences and journals, including, the *IEEE-Transactions on Computers*, *IEEE-Transactions on Parallel and Distributed Systems*, *IEEE- Transactions on Signal Processing*. His research interests include *parallel computing, cloud computing, and Big Data computing*. He is currently on the editorial boards of *the IEEE Transactions on Computers*, *International Journal of Pattern Recognition and Artificial Intelligence*. He is an outstanding member of CCF.

Keqin Li (Fellow, IEEE) is currently a SUNY distinguished professor of computer science with the State University of New York. He is also a national distinguished professor with Hunan University, China. He has authored or coauthored more than 810 journal articles, book chapters, and refereed conference papers. His research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing. He holds nearly 60 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top ten most influential scientists in parallel and distributed computing based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor for the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He was on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and the *IEEE Transactions on Sustainable Computing*. He was the recipient of the several best paper awards.