CoopEdge: Cost-effective Server Deployment for Cooperative Multi-Access Edge Computing

Rong Cong[1], Zhiwei Zhao[1,*], Linyuanqi Zhang [1] and Geyong Min[2,*]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[2]Department of Computer Science, University of Exeter, UK
{congrong, zhiwei, yuanqi}@mobinets.org; g.min@exeter.ac.uk

***Abstract*—The combination of 5G and edge computing has been envisioned as a promising paradigm to empower pervasive and intensive computing for the Internet-of-Things (IoT). High deployment cost is one of the major obstacles for realizing 5G edge computing. Most existing works tried to deploy the minimum number of edge servers to cover a target area by avoiding coverage overlaps. However, following this framework, the resource requirement per server will be drastically increased by the peak requirement during workload variations. Even worse, most resources will be left under-utilized for most of the time. To address this problem, we propose CoopEdge, a cost-effective server deployment scheme for cooperative multi-access edge computing. The key idea of CoopEdge is to allow deploying overlapped servers to handle variable requested workloads in a cooperative manner. In this way, the peak demands can be dispersed into multiple servers, and the resource requirement for each server could be greatly reduced. We further propose a two-step incremental algorithm, which jointly decides the server deployment and cooperation policies for fast convergence. We evaluate CoopEdge based on seven real-world edge applications. The results show that compared with the state-of-the-art works, CoopEdge significantly reduces the deployment cost by 38.7% and improves resource utilization by 36.2%.**

***Index Terms*—Cooperative edge computing, Cost-effective, Server placement, 5G edge network**

I. Introduction

The recent advances in 5G communication technologies have led to the proliferation of computing-intensive and latency-sensitive applications, such as real-time video analysis, automated driving, VR/AR, etc [1], [2]. Edge computing has emerged as a novel paradigm to satisfy the stringent delay-and-privacy requirements of such applications. In this way, Internet-of-Things (IoT) devices can offload their computing tasks to nearby edge servers [3]. While 5G edge computing is promising, the infrastructure deployment remains one of the the fundamental and critical challenges for network operators, of which the projected value of IoT devices is estimated to be as high as $30.9 billion in 2025 [4].

Server deployment refers to deploying a number of edge servers to cover IoT devices in a target area such as smart buildings, communities and factories [5]–[7]. Specifically, given IoT devices and their workloads, network operators need to decide the number of edge servers, the resources of each server and the deployment locations to handle the computational requests within the target area. Figure 1 illustrates the server deployment scenario for a smart factory. We need to deploy a number of edge servers, with appropriate resources and locations, to serve the heterogeneous and geographically distributed IoT requests in the factory area.

Most existing works tried to deploy the minimum number of edge servers to cover the target area by avoiding coverage overlaps of different servers [7]–[9]. Each IoT device is assigned to a dedicated server in the deployment plan. The resource requirement of a server is then determined by the sum of the IoT workloads within its coverage. For example, the state-of-the-art works [7], [8] tried to deploy the minimum number of edge servers considering wireless and traffic diversities of the IoT devices. *These works assumed stable workloads from the IoT devices, which can lead to a significant increase to per-server resource requirement and the overall deployment cost, due to the workload variations in the real-world scenarios.* The reason is that, in 5G edge computing, the difference between peak resource demands and the average demands can be very large [10]–[12]. When all servers are equipped with the amount of resource for the peak demands, more resources will be left under-utilized for most of the time.

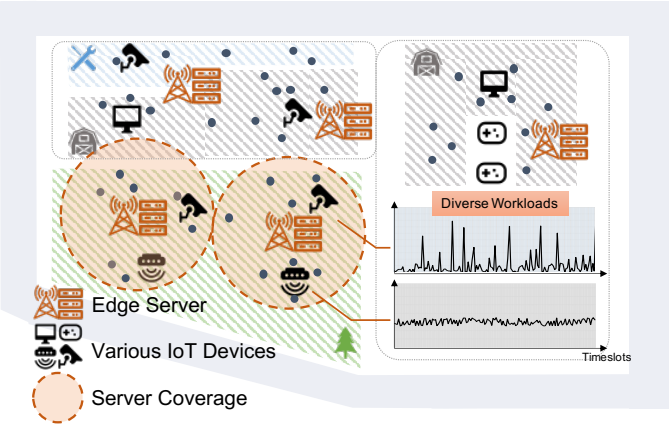


Fig. 1: Edge server deployment in a smart factory.

To address the problem, an intuitive idea is to deal with the peak workloads by using multiple servers. By distributing the peak workloads to multiple servers, the per-server resource requirement can be reduced. Some existing works tried to reduce the deployment cost following this idea. Zhao et al. [13] established on/off switches for the deployed servers. By turning off some servers when the workload is low, the energy

consumption can be reduced. Nevertheless, the performance in terms of deployment cost is still not optimized. Wang et al. [6] and Zhao et al. [13] used multi-server load balancing/scheduling to deal with workload variations. However, these schemes still aim at minimizing the number of servers by using fixed IoT-server assignment, which results in two important limitations: 1) limited multi-server opportunity for handling the peak workloads in a cooperative manner; and 2) limited reduction for per-server requirement due to the fixed IoT-server assignment.

Different from the above works that consider optimizing resource utilization *after* the server deployment, in this paper, *we intentionally exploit the multi-server cooperation during the deployment process by allowing overlapping server placements*. Our basic idea is to trade server coverage for more the cooperation opportunities among multiple servers, which could drastically decrease the resource requirement per server as well as the overall deployment cost. This idea faces two challenges as follows.

- More overlapping servers are expected to reduce more resource requirement for each server but increase the number of servers required for the full service coverage. We need to find a good trade-off between coverage and the cooperation opportunities.
- Existing works explicitly assign IoT devices to servers, such that the deployment cost can be evaluated. With overlapping server placements, the IoT-server assignment becomes implicit, i.e., an IoT device can be covered by different servers when its workload varies. This change leads to a coupled problem of workload allocation and finding deployment positions. More specifically, the server deployment policies determine the decision space of workload allocation, and in return, the workload allocation affects the deployment cost of different policies.

To address the above challenges and achieve cost-effective deployment, we propose CoopEdge, a novel server deployment scheme for cooperative multi-access edge computing. CoopEdge has three key innovations. First, CoopEdge intentionally introduces overlapping server placement, which could exploit the multi-server cooperation to reduce the per-server resource requirement. Second, we propose a two-step incremental deployment algorithm which gradually finds a good trade-off between server cooperation and number of required servers, such that the overall deployment cost is expected to be minimized. Third, we jointly optimize the server deployment and workload allocation, by employing an accurate potential estimation for multi-server cooperation opportunities. We conduct extensive measurement study and simulation experiments with seven real-world edge applications with varying workloads [14]–[20]. The evaluation results show that compared to the state-of-the-art works, CoopEdge can significantly reduce the overall deployment cost and improves the resource utilization.

The major contributions of this paper are summarized as follows:

- We propose CoopEdge, a cost-effective server deployment scheme for cooperative multi-access edge computing, which exploits coverage overlaps to reduce per-server resource requirement and overall deployment cost.
- We model the joint optimization problem of server deployment and workload allocation as a mixed-integer nonlinear programming problem, which accurately characterizes the positive and negative impacts of the server cooperation on server deployment.
- We design a two-step incremental deployment algorithm, which decouples the two sub-problems and obtains a great trade-off between multi-server cooperation and service coverage.
- We evaluate CoopEdge based on seven real-world applications. The results show that CoopEdge can significantly reduce the overall deployment cost by 38.7% and improve the resource utilization by 36.2%.

The rest of this paper is organized as follows. Section II further illustrates the core idea of CoopEdge and gives the problem formulation. Section III describes the two-step incremental deployment algorithm. Section IV presents the experiment results and the corresponding analysis. Section **??** reviews the related work. Section VI concludes this paper.

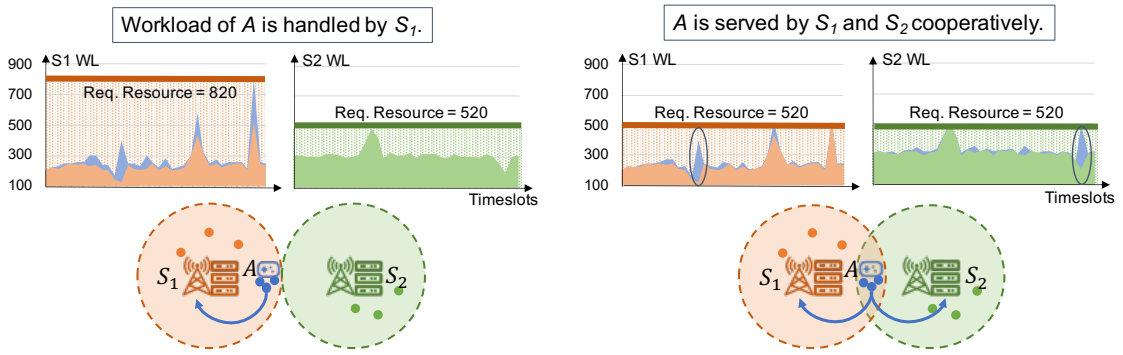
II. Design and Problem Formulation

This section will further illustrate the core idea of CoopEdge using a case scenario. Then we will present the system model, the corresponding assumptions and the problem formulation.

A. Design Overview

Figure 2 shows an example to illustrate the core idea of CoopEdge. In the case scenario, devices in an AR game room keep generating varying workloads of image processing. The diagrams show the varying workloads for both server S_1 and S_2, whose covered devices and their workloads are marked as orange and green colors, respectively. Specifically, the AR workloads are denoted with blue curves. Figure 2(a) shows the traditional deployment approach without coverage overlaps. To maximize server coverage, device A is assigned to server S_1. The resources required by S_1 depends on the peak of aggregated IoT workloads, which is the sum of A peaks and other covered nodes. Figure 2(b) shows an another approach which deploys S_1 and S_2 with overlap to serve device A cooperatively. Compared to the first approach, it can significantly *reduced the resource requirements of servers* by around 30% (from 1340 to 1040). The reason is that the latter approach leverages the time interleaving of peak workloads of S_1 and S_2 and thus can handle variable demands cooperatively. It is noteworthy that the cooperation approach reduces the per-server resource requirement *at the cost of coverage shrinking* due to the overlapping server placement. Apparently, the cooperation deployment scheme needs more edge servers to fully cover a large-scale target area. Hence, it is critical to find a great trade-off between coverage and cooperation opportunities in CoopEdge.

In this paper, we try to deploy cooperative edge servers for cost-effective deployment to cover all IoT devices and



(a) Deploying edge servers without server cooperation. (b) Deploying edge servers with server cooperation mechanism.

Fig. 2: An illustration of server cooperation mechanism in CoopEdge.

fulfill their computation demands in a target area. With the help of 5G edge computing, IoT devices can offload their computation-intensive and latency-sensitive tasks (e.g., AR/VR games and real-time video analysis) to edge servers. Here, the computation demands of IoT devices are diverse and time-varying. Even for the same application requests, the computation demands are still distinct due to network fluctuations and the difference of input data [21]. We assume that offloaded tasks are data-parallel, i.e., computation workloads can be divided into several partitions and offloaded to multiple edge servers for parallel processing [22].

B. System Model

TABLE I: Main parameters and their descriptions in this paper.

Parameter	Description
N	The number of IoT devices in the target area.
S_j	Edge server j.
$\mathcal{N}_j$	Set of IoT devices covered by S_j
M	The number of deployed servers.
$\mathbf{T}$	The set of time slots during the operation process.
w_i^t	Workload of device i in time slot t.
l_i	Position of device i.
γ	The communication radius of edge server.
p_j	The deployed position of server j.
r_j	The allocated resources of server j.
Δ	The size of unit deployed resource.
δ	The number of incremental server in the TID algorithm.
$a_{i,j}^t$	Workload assignment of device i to S_j in time slot t.
C^I	Infrastructure cost per server.
C^R	Resource cost per resource unit.
ϕ_j^t	Aggregated workloads of S_j in time slot t.

The main parameters in our model and their descriptions are listed in Table I. In a target area with N IoT devices, each device generates resource-hungry tasks with strict Quality-of-Service (QoS) requirements and offloads them to nearby edge servers. The location of device i is denoted as l_i. Assume that the whole operational time $\mathbf{T}$ can be divided into T discrete time slots. The workloads of device i can be denoted as $\mathbf{W}_i = \{w_i^1, w_i^2, ..., w_i^T\}$, where w_i^t is i's workload in time slot t. With the goal of minimizing the overall deployment cost, we jointly optimize the server deployment and the cooperation policies, and the concrete determined parameters are modelled as follows.

Server deployment. Server deployment decisions include the number of edge servers M, server resources $\mathbf{R} = \{r_j\}_{:j=1,...,M}$ and their corresponding positions $\mathbf{P} = \{p_j\}_{:j=1,...,M}$. Without loss of generality, edge servers are equipped with a number of standardized computing units like rack servers. Resource allocation indicator r_j is an integer variable satisfying $r_j = k\Delta$, where Δ is the size of a resource unit and k is a non-negative integer, i.e., $k \in \{0, 1, ...\}$.

Cooperation policy. In CoopEdge, variable device workloads can be cooperatively distributed to multiple edge servers. In this way, the server cooperation, i.e., workload assignment, becomes critical yet inevitable for cost-effective deployment. Let $\mathbf{A} = \{a_{i,j}^t\}_{:i\in N, j\in M, t\in \mathbf{T}}$ be the set of workload assignment policies. The assignment for device i's workload in time slot t is denoted as $a_{i,j}^t \in [0, 1]$.

Overall deployment cost. The overall deployment cost consists of two parts. 1) Infrastructure cost, which is determined by the number of edge servers, includes server hardware cost (e.g., rack and antennas) and installation cost. The infrastructure cost of one server is denoted as C^I. 2) Resource cost, i.e., the cost of server computing resource. The cost of one resource unit is denoted as C^R.

C. Problem Formulation

Based on the above parameters, we can formulate the server deployment in CoopEdge as a mixed-integer nonlinear programming problem.

As shown in Eq.(1), our goal is to minimize the overall deployment cost including infrastructure cost and resource cost by optimizing 1) the number of edge servers (M), 2) server positions ($\mathbf{P}$), 3) resources allocated to each server ($\mathbf{R}$) and 4) workload assignment with cooperation manner ($\mathbf{A}$). Constraint $\mathcal{C}_1$ ensures all IoT devices in the target area can be covered by deployed edge servers, whose coverage is modelled as a circle with γ radius. Constraint $\mathcal{C}_2$ indicates the sum of assigned workloads should not exceed the server resource. Constraints

$\mathcal{C}_3$ and $\mathcal{C}_4$ jointly ensure device workloads in any time slot can be entirely handled by the corresponding servers.

$$\mathcal{P}: \min_{M,\mathbf{P},\mathbf{R},\mathbf{A}} \ C^I M + \sum_{j=1}^{M} C^R r_j \tag{1}$$

$$s.t. \mathcal{C}_1: \sum_{j=1}^{M} \max\{\gamma - ||l_i - p_j||, 0\} > 0, \ \forall i \in \{1,..,N\} \tag{2}$$

$$\mathcal{C}_2: \sum_{i \in \mathcal{N}} a_{i,j}^t w_i^t \leq r_j, \ \forall j \in \{1,..,M\}, \forall t \in \mathbf{T} \tag{3}$$

$$\mathcal{C}_3: a_{i,j}^t \in [0,1], \ \forall i,j,t \tag{4}$$

$$\mathcal{C}_4: \sum_{j=1}^{M} a_{i,j}^t = 1, \ \forall i \in \{1,..,N\}, \forall t \in \mathbf{T} \tag{5}$$

However, it is a non-trivial task to solve the above problem. The first major challenge is the interrelationship among the determined parameters. More precisely, the server deployment policies decide the solution space of workload assignment, while the optimal server deployment needs to leverage the cooperation opportunities which are affected by workload assignment. Second, given the server deployment policies, optimizing one-timeslot workload assignment is still a mixed-integer programming problem that is difficult to solve. Considering workload assignment needs to be decided in almost every time slot, it is critical and challenging to design a light-weighted and efficient approach.

III. Algorithm Design

A. Algorithm Overview

To address the above challenges, we propose a Two-step Incremental Deployment (TID) algorithm, as shown in Figure 3. Similar to the existing deployment works [7], [23], the algorithm first discretizes the continuous target area into a number of discrete candidate positions $\mathbf{P}^{Cand}$. Distinct from the above works that used peak workloads as the area-discretization criteria, we divide area by the average workload and evaluate candidates' cooperation opportunities by the optimized potential and cooperation ability. Then, we execute the TID algorithm, which uses incremental method to trade off coverage and cooperation opportunities and iteratively updates

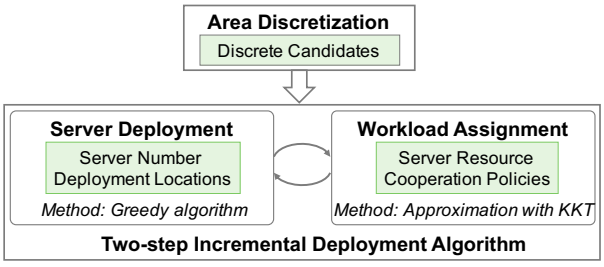


Fig. 3: Overview of the proposed two-stage incremental deployment algorithm.

Algorithm 1: Two-step Incremental Deployment (TID)

Input: N, Set of candidate positions $\mathbf{P}^{Cand}$.
Output: M, $\mathbf{P}$, $\mathbf{R}$, $\mathbf{A}$.

1 Deploy minimum servers to get full-node coverage;
2 Initialize M, $\mathbf{P}$, $\mathbf{R}$ and $\mathbf{A}$;
3 $M^{Opt} = M$, $\mathbf{P}^{Opt} = \mathbf{P}$, $\mathbf{R}^{Opt} = \mathbf{R}$ and $\mathbf{A}^{opt} = \mathbf{A}$;
4 Calculate deployment cost C^{Opt};
5 Calculate the upper bound of server number M^{Up};
6 **while** $M \leq M^{Up}$ **do**
7 **for** i = *1,2,...,*$Iter^{max}$ **do**
8 **forall** $S_j \in \mathbf{S}_\delta$ **do**
9 Select $\mathbf{S}_j$ with the max. optimization potential using Eq. (6);
10 Deploy S' with the maximal cooperation ability of S_j using Eq. (7);
11 Based on $\mathbf{P}$, obtain $\mathbf{R}$, $\mathbf{A}$ and C using Algorithm 2;
12 $\mathbf{P} = \mathbf{P} \cup \{p_{S'}\}$;
13 **end**
14 **if** $C < C^{Opt}$ **then**
15 $C^{Opt} = C$, $M^{Opt} = M$, $\mathbf{P}^{Opt} = \mathbf{P}$, $\mathbf{R}^{Opt} = \mathbf{R}$, $\mathbf{A}^{Opt} = \mathbf{A}$;
16 **end**
17 **end**
18 $M = M + \delta$;
19 **end**

decisions to decouple the server deployment and workload assignment, as shown in Algorithm 1.

The TID algorithm gradually adds the number of edge servers from the minimum number of servers to find the optimal deployment policy with the minimum overall cost. In each round, it updates the server deployment and workload assignment policies iteratively to decouple the two sub-problems. Specifically, the deployment process starts from the policy that tries to fully cover the target area with the minimum number of servers (Line 1 to 2). With gradually increasing the number of servers, the algorithm iteratively updates the coupled decisions to minimize the overall deployment cost, until the server number reaches M^{Up} (Line 4 to 17). Note that server-number upper bound M^{Up} here is set as the number of IoT devices, which means one device will be covered by only one server. The details of the coupled sub-problems are described in the next subsections.

The basic steps of server deployment in the TID algorithm with the fixed server number is shown in Line 6 to 10. Considering the hardness of directly placing δ edge servers $\mathbf{S}_\delta$ into the target area, we repeat the single-server deployment for δ times. In each incremental process, we first select one deployed server S_j to be further cooperated and then add server S' at the appropriate position. The first step is to reduce the searching space of the added server position from the whole area to the candidates near S_j. After narrowing the selection space, the added server should be placed at the

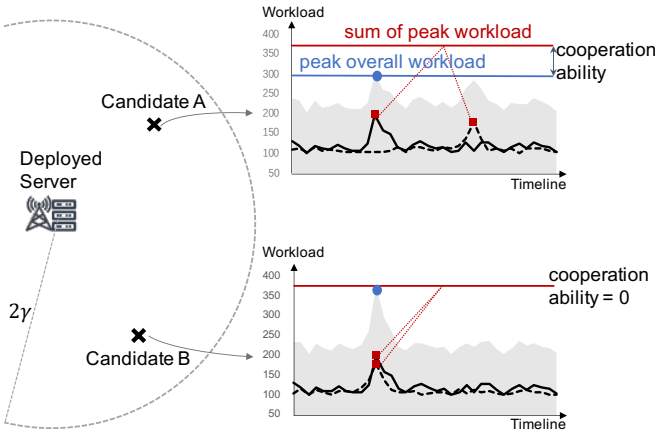


Fig. 4: An illustration of candidates' cooperation ability.

candidate position with the maximum cooperative potential to reduce the per-server resource requirement.

The key problems in the above deployment process are 1) which deployed server should be selected to be helped, and 2) where the added server should be placed. Intuitively, the added server should be deployed around the server which can benefit a lot via cooperation. To quickly evaluate the cooperation benefits of deployed servers, we propose the optimization potential metric. Besides, we estimate the cooperation ability to numerically characterize the impact of the incremental server without any information on workload assignment.

Optimization potential, i.e., the selection criteria of deployed server. The optimization potential of deployed edge server S_j is the resource reduction opportunity via cooperation, which depends on the burst demand of S_j and the number of connection. Therefore, in line 9, we use the difference between the peak and the average workload to estimate the optimization potential of server S_j, as shown in Eq. (6).

$$\max_{t\in\mathbf{T}} \sum_{i=1}^{N} a_{i,j}^t w_i^t - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} a_{i,j}^t w_i^t \tag{6}$$

Cooperation ability, i.e., the criteria of candidate position evaluation. As shown in line 10, we then estimate the cooperation ability of the incremental server S'. Eq. (7) gives the calculation of the cooperation ability of S' located in $p_{S'}$, i.e., the difference between the sum of peaks of covered devices and the server aggregated peak.

$$\sum_{i\in\mathcal{N}_{S'}} \max_{t\in\mathbf{T}} w_i^t - \max_{t\in\mathbf{T}} \sum_{i\in\mathcal{N}_{S'}} w_i^t \tag{7}$$

where $\mathcal{N}_{S'}$ is the set of IoT devices covered by S'.

Figure 4 shows an illustration of candidates' cooperation ability. Given the deployed server, there are two candidates (A and B) around the server. We want to find the candidate position to deploy the incremental server which fully leverages the cooperation opportunity. Recall that CoopEdge reduces the per-server resource requirement by handling the burst demands generated in different time slots. Candidates A and B have the same node-peak value in time slots, marked as red squares. We see that the sum of node peaks are same for the two candidates (denoted with red lines), while their aggregated peak workloads are distinct (denoted with blue lines). When most of IoT devices generate heavy demands *at the same time*, the deployed server will get little benefit from the incremental server cooperation. As a result, we estimate the cooperation ability of candidates using the difference between the sum of node peaks and the server aggregated peak.

Algorithm 2: Workload Assignment in TID algorithm.

Input: N, M, $\mathbf{P}$.
Output: $\mathbf{A}$, $\mathbf{R}$.

1 Initialize the set of optimized time slots $\mathbf{T}^{Opt} = \varnothing$;
2 **for** $iter = 1, ..., iter^{Rand}$ **do**
3 Randomly select a deployed server S_j to optimize;
4 Obtain k time slots $\mathbf{T}^k$ with k-top workload of S_j;
5 $\mathbf{T}^{Opt} = \mathbf{T}^{Opt} \cup \mathbf{T}^k$;
6 **end**
7 **forall** *time slot* t *in* $\mathbf{T}^{Opt}$ **do**
8 Obtain $\mathbf{A}^t$ by solving problem $\mathcal{P}_2$ with KKT conditions;
9 **end**
10 Obtain $\mathbf{R}$ based on $\mathbf{A}$;

B. Workload Assignment

Given the deployed servers, we need to optimize the workload assignment via the cooperation manner to reduce the per-server resource requirement. But even the one-timeslot workload assignment is still a non-convex problem. To address this challenge, we use the approximation method transforming the non-convex problem into a convex optimization problem and then use the Karush-Kuhn-Tucker (KKT) conditions to solve it. The second step of TID, workload assignment, is summarized in Algorithm 2.

The algorithm selects a part of time slots from $\mathbf{T}$ to optimize the cooperation policy and adds them into $\mathbf{T}^{Opt}$, i.e., the set of optimized time slots (Line 1). Considering some edge servers may have burst workloads in the same time, we use the set structure to record the optimized time slots, aiming to avoid optimizing workload assignment in the same time slots multiple times. Then, the algorithm randomly samples edge server S_j to further reduce its resource requirement by server cooperation (Line 2 to 6). For each selected server, we find its k-top workloads and add the corresponding time slots into $\mathbf{T}^{Opt}$ (Line 4 to 5). In this way, the continuous-time optimization problem is converted into a workload assignment problem in one time slot. Given optimized time slot $t_o \in \mathbf{T}^{Opt}$, the sub-problem of workload assignment in time slot t_o can be formulated as

$$\mathcal{P}_1 : \min_{\mathbf{A}^{t_o}} \sum_{j=1}^{M} \max_{t\in\mathbf{T}} \Big(\sum_{i=1}^{N} a_{i,j}^t w_i^t\Big) \tag{8}$$

$$s.t.\ \mathcal{C}_3, \mathcal{C}_4 \tag{9}$$

Here, $\max_t(\sum_{i=1}^{N} a_{i,j}^t w_i^t)$ indicates the peak workload of server j. It means that the optimization objective is the sum of required resource, i.e., the sum of server's peak workload.

However, the objective of $\mathcal{P}_1$ remains a hard-solving non-convex function. This paper uses the approximation optimization method to solve it. Specifically, we approximate the problem using the following *log-sum-exp* approximation [24].

$$\max_{t\in\mathbf{T}} \phi_j^t \leq \frac{1}{\zeta} log(\sum_{t\in\mathbf{T}} exp(\zeta\phi_j^t)) \tag{10}$$

where $\phi_j^t \triangleq (\sum_{i=1}^{N} a_{i,j}^t w_i^t)$ denotes the aggregated workloads of S_j in time slot t. With this approximation function, problem $\mathcal{P}_1$ can be approximated to the following problem $\mathcal{P}_2$ for large ζ.

$$\mathcal{P}_2 : \min_{\mathbf{A}^{t_o}} \frac{1}{\zeta}\sum_{j=1}^{M} log(\sum_{t\in\mathcal{T}} exp(\zeta\phi_j^t)) \tag{11}$$

Now, we can derive the optimal solutions to problem $\mathcal{P}_2$ using the KKT conditions. Let $\Lambda = \{\lambda_1, ..., \lambda_N\}$ be the Lagrange multiplier associated with constraints $\sum_{j=1}^{M} a_{i,j}^{t_0} = 1,\ \forall i \in \mathcal{N}$.

The corresponding Lagrange function is

$$\begin{aligned} L(\mathbf{A}^{t_o}, \lambda) &= \frac{1}{\zeta}\sum_{j=1}^{M} log(\sum_{t\in\mathbf{T}} exp(\zeta\phi_j^t)) + \sum_{i=1}^{N}\lambda_i(\sum_{j=1}^{M} a_{i,j}^t - 1) \\ &= \frac{1}{\zeta}\sum_{j=1}^{M} log(\sum_{t\in\mathbf{T}} exp(\zeta\phi_j^t)) + \Lambda\mathbf{A}^{t_o} \end{aligned} \tag{12}$$

Then the KKT conditions can be inferred as follows.

$$\frac{d}{d\mathbf{A}^{t_o}}\frac{1}{\zeta}\sum_{j=1}^{M} log(\sum_{t\in T} exp(\zeta\phi_j^t)) + \Lambda = 0 \tag{13}$$

$$\sum_{j=1}^{M} a_{i,j}^t = 1\ , \forall i \in \{1, .., N\},\ t \in \mathbf{T} \tag{14}$$

$$\lambda_i \geq 0\ , \forall i \in \{1, .., N\} \tag{15}$$

IV. Evaluation

This section introduces the setup used for the experiments with the measurement of seven real-world benchmark applications in edge computing to simulate variable workloads. Then, the performance of CoopEdge is evaluated in terms of the overall deployment cost, the number of servers and resource utilization, compared with the state-of-the-art deployment approaches.

A. Experiment Setup

1) Settings: We consider a 3km $\times$ 3km edge network with 100-500 IoT devices placed as a Poisson point process model [28] and each IoT device generates dynamic computational tasks following a Poisson process. The workloads of each task are set based on the measurement results of real-world applications, which is detailed in Section IV-A2. We tend to deploy edge servers to cover all IoT devices in the target network and allocate the corresponding resources to handle their computation demands. Edge server is customized with a number of standardized resource units equipped with Intel Xeon E5-2637 (3.0GHz) and 5TB storage capacity, according to the parameters of Dell PowerEdge R720 [29]. We thus set the unit resource cost C^R as \$1399.00. The deployment cost per server C^I is set to \$700.00 according to the price of Huawei outdoor marcro base station BTS3900A [30]. Each server uses LTE/5G/6G telecommunication technology with 50m to 1km communication radius [31].

2) Measurement Dataset: To better simulate the computation workloads of heterogeneous IoT devices, we first measure seven real-world applications and then randomly generate variable workloads of different devices. These measured applications comprise object detection, natural language processing, image classification and VR games, which are regarded as edge benchmarks in many related schemes [1], [32]. The whole measurement experiments are implemented on two edge platforms: NVIDIA Jetson TX2 and HP Omen with a GPU GeForce RTX 2070Ti. Each experiment is repeated 10 times for 10 minutes with the fixed configurations, i.e., frame rate 30FPS and resolution 720p. It is worth noting that we use TFLOPS instead of CPU cycles as the workload metric since some edge applications need both GPU and CPU resources to improve processing efficiency, like AR/VR games and vision-based video analytics [33], [34]. TFLOPS can be obtained by the data collector in the performance monitor provided by Windows 10. TABLE II shows the measurement results.

Based on the measurement workloads of different applications, we generate the diverse and variable computation demands of IoT devices. Specifically, the generation of computation tasks follows a Poisson process with an arrival rate uniformly distributed in [0.05, 0.2]. For each computational task, the size of input data, i.e., video length, is modelled as a heavy-tailed distribution, as proven in the previous work [35].

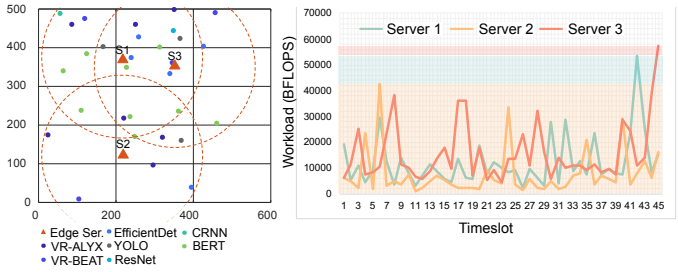
3) Benchmark and Metrics: We compare our deployment scheme with the state-of-the-art works [7], [8], which tend to deploy the minimum number of edge servers to cover the target network. For ease of explanation, the above works are roughly represented as SeparatEdge. In SeparatEdge, edge servers handle the assigned workloads independently with sufficient resources to satisfy the peak demands of covered IoT devices. With the assumption of static computation workloads, SeparatEdge discretizes the target area depending on node peak traffic.

Three key metrics are used to evaluate the performance of the above deployment works.

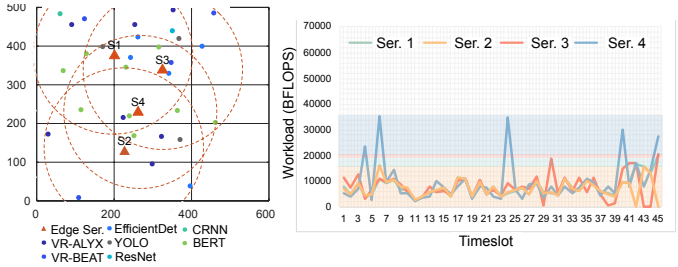
- Overall deployment cost, i.e., the sum of infrastructure cost and resource cost, is impacted by both the number of edge servers and their resource requirements. This metric is to figure out whether CoopEdge can achieve the great trade-off between coverage and the cooperation opportunities.
- The number of edge servers, used to fully cover the target area with IoT nodes in different deployment approaches.

TABLE II: Characteristics of benchmark applications and their workloads(TFLOPS) per unit time.

Model	Task Description	Dataset	Workloads	Setting
YOLOv5 [14]	Object detection	CoCo [25]	284.90	10min., 30fps, 720p
ResNet152 [15]	Image classification	CoCo [25]	187.70	10min., 30fps, 720p
CRNN [16]	Handwriting recognition	MNIST [26]	576.54	10min., 30fps, 720p
Efficient Det-7 [17]	Object detection	CoCo [25]	175.60	10min., 30fps, 720p
BERT [18]	Question answering	SQuAD v2.0 [27]	51.00	4 paragraphs about 50 questions
Half-Life: Alyx [19]	VR game	Steam	4162.79	10min.
BEAT SABER [20]	VR game	Steam	2731.29	10min.



(a) Deploy edge servers using SeparatEdge.



(b) Deploy edge servers using CoopEdge.

Fig. 5: A small-scale case: server placement (left) and their workloads (right) in different deployment schemes.

This metric is to evaluate the cost of increased edge servers in CoopEdge.

- Resource utilization, in terms of the average ratio of computation demands and server resources in each time slot.

B. Performance Evaluation

Based on the measurement of edge application workloads, we conduct extensive simulations with the above deployment approaches and compare their performance to obtain system insights. We first try to deploy edge servers for a small-scale case area and analyze the performance difference using the visual results. We further explore various practical scenarios by tuning related parameters including server coverage, network density, the average task arrival rate, network scale etc.

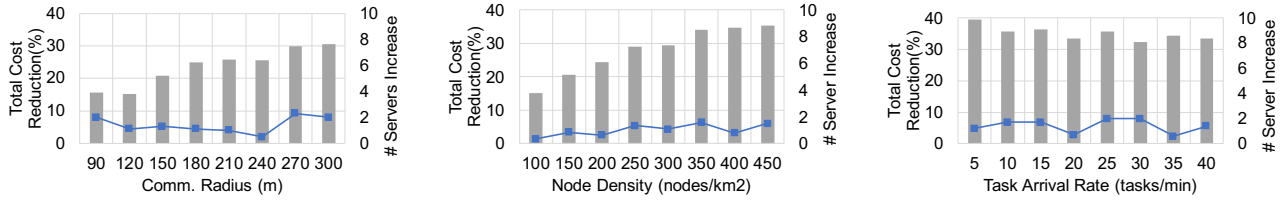
Case Study. Given a 500m $\times$ 600m area with 30 IoT devices, we tend to deploy a number of edge servers to fully cover the target area. Here, "fully cover" means the computation workloads from all covered devices can be fulfilled by server resource. The devices are heterogeneous and assume to generate computation tasks from the above seven real-world applications in a Poisson process, as marked in different color nodes.

Figure 5 shows the server deployment results (right sub-figure) and the corresponding workloads (left sub-figure) in CoopEdge and SeparatEdge. We see that 1) CoopEdge uses more edge servers (four) than SeparatEdge (three) to achieve full-service coverage, as our scheme pursues extending coverage overlaps to exploit more cooperation opportunities. Note that server positions are not identical in the two schemes due to the criteria difference in area discretization. CoopEdge discretizes area depending on the average workloads, whereas SeparatEdge makes the node peak workloads as the criteria. 2) Although CoopEdge tends to deploy more edge servers, the resource requirement are drastically decreased by 40.05% (from 153,352 to 91,931 BFLOPS), as marked in the corresponding color bars. The reason is that burst workloads are cooperatively handled by relatively idle servers. As plotted in the right sub-figures of server workloads, servers in SeparatEdge often have huge workload gap in the same time, while servers in CoopEdge almost entirely have the peak value, except Server 4 in CoopEdge. The reason is that some devices (AR games) with highly variable workloads are only covered by Server 4.

Deployment Cost. Figure 6 shows the performance comparison between CoopEdge and SeparatEdge in terms of the reduction in overall deployment cost and the increased number of edge servers. Specifically, the deployment scenarios are explored by tuning the parameters of the server communication range, node density and the task generation rate. Each of the simulation is repeated 100 times to reduce the negative impact of randomness.

Figure 6(a) shows the results with varying the communication radius of edge servers corresponding to different communication technologies. We see that 1) CoopEdge consistently outperforms SeparatEdge and reduces the overall deployment cost by 15% to 30%, especially in large-coverage networks. This is because the larger communication range leads to more server cooperation opportunities to share the burst workloads and thus improve the system performance. 2) CoopEdge deploys only $0.8 - 2.7$ more edge servers to extend the server overlaps for better cooperation. It means that CoopEdge achieves a satisfying trade-off between the coverage scales and the cooperation chances.

Figure 6(b) depicts the performance difference under different node densities with the fixed network scale (the number of IoT devices increases accordingly). We see that CoopEdge remains to outperform SeparatEdge, and the performance gap



(a) The performance difference with varying communication radius of edge server.

(b) The performance difference with varying network density.

(c) The performance difference with varying average request rate.

Fig. 6: The reduction on overall deployment cost and the increase of server number with different deployment schemes.

increases as node density increases. The probability of node burst overlapping increases when the node density increases. Figure 6(c) shows the results under the two deployment schemes with varying the average request generation rate. Recall that task generation of IoT devices is modelled as a Poisson process, whose arrival rate is the parameter tuned in this simulation. We observe that the reduction level does not change with the request rate increasing. It means that CoopEdge can efficiently tackle the diversity of workload variation without loss of system performance.

V. Related Work

This section will review the existing works of edge server deployment from two classifications and discuss their differences with CoopEdge.

A. Novel characteristics of 5G edge computing.

The fifth-generation (5G) network has been envisioned to provide low-latency connections and developed to markets in 2020 [36]. Although 5G edge computing is a promising paradigm for ubiquitous IoT devices, infrastructure deployment is still a challenging problem for the following characteristics of 5G edge computing. First, the communication radius of 5G servers decreases from 1-25 kilometres (4G) [37] to 100-300 metres [38], which leads to fewer IoT devices in server coverage. Second, the IoT systems tend to be massive in device density, network scale and computation workload. According to Cisco, the number of IoT connections will grow from 1.2 billion in 2018 to 4.4 billion by 2023 [39]. Third, the difference between peak resource demands and the average demands will be very large, especially video/graphic tasks such as AR/VR service [19] and object detection [10]. As a result, 5G edge servers are more sensitive to the workload changes of connected devices.

B. Deployment works for stable workloads.

Recent studies tried to deploy edge servers in various environments such as large-scale networks [7], intelligent production lines [6], city dense areas [40]. However, most works hold an assumption that node workloads are stable and used the sum of peak workloads as computation demands. For 5G edge networks, Li *et al.* [40] studied the edge server placement with the consideration of 5G User Plane Functions (UPF) and their effect on end-to-end delay. To maximize the profit of network operators, the authors designed a particle swarm optimization based algorithm to place edge servers with the guarantee of access latency. For the area division in edge server placement, Mathieu *et al.* [41] formulated the problem as a mixed integer linear programming problem and proposed a graph-based algorithm to find a near-optimal partition policy. For large-scale IoT systems, Zhao *et al.* [7] proposed a three-phase deployment approach that considers both the wireless diversity and traffic diversity of IoT devices to reduce the number of required edge servers. While the above deployment approaches can reduce the deployment cost to some extent, the assumption of stable workloads will lead to significant increase to per-server resource requirement and the overall deployment cost due to the workload variations in the real-world scenarios.

C. Deployment works for variable workloads.

To address the above problem, some works tried to deal with node peak demands using multiple servers. For example, Zhao *et al.* [13] combined the server deployment with the on/off switch of server status to deal with the variable workloads of end devices. The on/off switch manner tended to turn off some deployed servers when the workload was low, and thus the energy consumption can be reduced. Load balancing is also a promising mechanism to handle node burst workloads. Lahderanta *et al.* [5] and Wang *et al.* [6] focused on deployment problem in large-scale cities and smart factories, respectively. Specifically, Lahderanta *et al.* [5] proposed a PACK algorithm based on K-means to jointly optimize the server deployment and workload sharing. Wang *et al.* [6] presented a server deployment and load balancing approach to minimize the response time based on the space-time characteristics of industrial IoT (IIoT) nodes. Although these works can reduce the per-server resource requirement for time-varying computation demands, they still aimed at minimizing the number of servers with fixed IoT-server assignment. Distinct from the above works, CoopEdge initiatively allows overlapping server deployment and flexible IoT-server assignment to sufficiently exploit the multi-server cooperation opportunities in multi-access networks.

VI. Conclusion

In this paper, we propose CoopEdge, a cost-effective server deployment scheme for cooperative multi-access edge computing. Different from most existing works that tried to deploy

the minimum number of edge servers to the target area by avoiding coverage overlaps, CoopEdge exploits the multi-server cooperation by allowing overlapping server placement to handle device variable workloads with lower resource. The deployment problem in CoopEdge is formulated as a mixed-integer nonlinear programming problem and solved by the proposed two-step incremental deployment algorithm. Evaluation results show that CoopEdge can significantly decrease the overall deployment cost and improve the resource utilization.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No. 61972074 and No. 61972075) and the National Key Research and Development Program of China (No.2020YFE0200500). Zhiwei Zhao and Geyong Min are the corresponding authors.

REFERENCES

[1] G. Ananthanarayanan, V. Bahl, L. Cox, A. Crown, S. Nogbahi, and Y. Shu, "Video analytics-killer app for edge computing," in *Proceedings of the 17th annual international conference on mobile systems, applications, and services*, 2019, pp. 695–696.

[2] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[3] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, 2019.

[4] I. Analytics, "Internet of things (iot) and non-iot active device connections worldwide from 2010 to 2025 (in billions)," 2020.

[5] T. Lähderanta, T. Leppänen, L. Ruha, L. Lovén, E. Harjula, M. Ylianttila, J. Riekki, and M. J. Sillanpää, "Edge computing server placement with capacitated location allocation," *Journal of Parallel and Distributed Computing*, vol. 153, pp. 130–149, 2021.

[6] J. Wang, D. Li, and Y. Hu, "Fog nodes deployment based on space–time characteristics in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3534–3543, 2020.

[7] Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, and Q. Ni, "Deploying edge computing nodes for large-scale iot: A diversity aware approach," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3606–3614, 2018.

[8] T. K. Rodrigues, K. Suto, and N. Kato, "Edge cloud server deployment with transmission power control through machine learning for 6g internet of things," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 2099–2108, 2021.

[9] 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 Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 494–503, 2020.

[10] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, "Flexible high-resolution object detection on edge devices with tunable latency," in *Proc. of ACM MobiCom 2021*, pp. 559–572.

[11] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele, "Posetrack: A benchmark for human pose estimation and tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5167–5176.

[12] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proc. of ACM SIGCOMM 2020*, pp. 359–376.

[13] S. Zhao, X. Zhang, P. Cao, and X. Wang, "Design of robust and efficient edge server placement and server scheduling policies," in *Proc. of IEEE/ACM IWQOS 2021*, pp. 1–7.

[14] G. Jocher, K. Nishimura, T. Mineeva, and R. Vilariño, "Yolov5," *Code repository https://github. com/ultralytics/yolov5*, 2020.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of IEEE/CVF CVPR 2016*, 2016, pp. 770–778.

[16] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[17] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proc. of IEEE/CVF CVPR 2020*, pp. 10 781–10 790.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] "Alyx half-life 2," https://www.half-life.com/en/alyx/.

[20] "Beat saber," https://beatsaber.com/.

[21] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. of IEEE INFOCOM 2020*, pp. 257–266.

[22] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *Proc. of ACM MobiCom 2021*, pp. 201–214.

[23] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," *IEEE Transactions on Mobile computing*, vol. 9, no. 5, pp. 643–656, 2009.

[24] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE transactions on information theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," *arXiv preprint arXiv:1806.03822*, 2018.

[28] R. W. Heath, M. Kountouris, and T. Bai, "Modeling heterogeneous network interference using poisson point processes," *IEEE Transactions on signal processing*, vol. 61, no. 16, pp. 4114–4126, 2013.

[29] "Dell poweredge r720," https://www.dell.com/support/home/zh-cn/product-support/product/poweredge-r720/docs.

[30] "Cost of huawei marco base station bts3900a," https://e.huawei.com/en/products/wireless/gsm-r/radio-access-network/bts3900a.

[31] R. Cong, Z. Zhao, G. Min, C. Feng, and Y. Jiang, "Edgego: A mobile resource-sharing framework for 6g edge computing in massive iot systems," *IEEE Internet of Things Journal*, 2021.

[32] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "Defog: fog computing benchmarks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 47–58.

[33] Y. Xiang and H. Kim, "Pipelined data-parallel cpu/gpu scheduling for multi-dnn real-time inference," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 392–405.

[34] Q. Zhang, H. Sun, X. Wu, and H. Zhong, "Edge video analytics for public safety: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1675–1696, 2019.

[35] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar vbr video traffic," *ACM SIGCOMM computer communication review*, vol. 24, no. 4, pp. 269–280, 1994.

[36] G. Liu, Y. Huang, Z. Chen, L. Liu, Q. Wang, and N. Li, "5g deployment: Standalone vs. non-standalone from the operator perspective," *IEEE Communications Magazine*, vol. 58, no. 11, pp. 83–89, 2020.

[37] A. Jha and D. Saha, "Coverage and capacity dynamics in 4g-lte deployment in india," in *2019 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2019, pp. 1–8.

[38] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5g ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[39] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, 2020.

[40] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 55–67, 2021.

[41] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 787–796, 2018.