Optimized Edge Node Allocation Considering User Delay Tolerance for Cost Reduction

Xiaoyu Zhang, Shixun Huang, Hai Dong, *Senior Member, IEEE,* Zhifeng Bao, Jiajun Liu, and Xun Yi

Abstract—With the rise of 5G technology, Mobile (or Multi-Access) Edge Computing (MEC) has become crucial in modern network architecture. One key research area is the effective placement of edge nodes, which has attracted significant attention. Service providers strive to minimize deployment costs for these nodes within a network. Although many studies have explored optimal strategies for reducing these costs, most overlook the allocation of computational resources and the users' tolerance for delays. These factors add complexity, making previous methods less adaptable. In this paper, we define the Cost Minimization in MEC Edge Node Placement problem. Our goal is to find the optimal strategy for deploying edge nodes that minimize costs while cater to users' delay tolerance limits. We prove the NP-hardness of this problem and provide a range of solutions, including Cluster-based Mixed Integer Programming, Coverage First Search, and Distance-Aware Coverage First Search, to address this challenge effectively and efficiently. Additionally, we propose a fine-grained optimization approach for allocating computational resources to edge nodes based on user service requests, significantly lowering deployment costs. Extensive experiments on a large-scale real-world dataset show that our solutions outperform the state-of-the-art in efficiency, effectiveness, and scalability.

Index Terms—Mobile (or Multi-Access) Edge Computing, Edge node placement, Cost minimization, User delay tolerance

1 Introduction

Mobile (or Multi-Access) Edge Computing (MEC) emerges as a novel computing paradigm accompanying the rise of 5G technology. In MEC, a large number of servers with limited computing and storage capacity, known as edge servers or edge nodes, are deployed to the edge of a network in a distributed manner. Users with the MEC technology are closer to computing resources, which can not only significantly reduce their network latency but also provision them with substantial and nearby computing resources [1].

Problem. In this paper, we study the problem of optimal edge node deployment, aiming to provide qualified and low-latency services to massive mobile users citywide with minimum cost. There are three primary factors that require to be considered during the decision-making process of edge node deployment:

First, guaranteeing the Quality of Service (QoS) is a fundamental requirement of the network deployment [1]. For example, latency, as one of the most important QoS factors, should not exceed users' tolerance [2, 3, 4, 5, 6]. Second, minimizing the deployment cost is always prioritized and should not be neglected [7, 8, 9, 10]. There are many factors that can affect the cost, including edge node site selection, computation resource allocation, computation task assignment, etc. Third, resource allocation is another factor for MEC deployment, which is closely related to the two aforementioned factors. MEC is designed to mitigate the resource scarcity issue and should be optimized for higher productivity [11, 12, 13, 14, 15, 16, 17]. To avoid introducing extra network deployment costs, it is always preferred to allocate edge nodes an appropriate amount of computation resources that is able to not only satisfy the current user's QoS network requirements but also provide high-quality services for the foreseeable future utilization.

Motivation. There is always a trade-off between the edge node deployment cost and the delay experienced by mobile users [18]. Such a trade-off is highly related to edge node site selection and corresponding resource allocation. To be more specific, a longer distance between edge nodes and base stations would cause a longer transmission delay, while a larger workload from the base stations would also cause a larger computation delay. Deploying more edge nodes can potentially reduce the transmission delay by decreasing the average distance between base stations and edge nodes. Also, adding more servers (i.e. computing resources) into edge nodes can cut down the computation delay as edge nodes would have higher computation capacity. However, both cases will inflate the overall deployment cost. Fortunately, users would never expect a zero-latency network and instead have tolerance thresholds on the delay. Thus, by leveraging the threshold, we have an opportunity to optimize the edge node deployment strategy with the minimum overall deployment cost. The following sample scenario demonstrates how the edge node selection and the

- *Xiaoyu Zhang, Hai Dong (Corresponding Author), Zhifeng Bao and Xun Yi are with the School of Computing Technologies, RMIT University, Melbourne, VIC, Australia*
  E-mail: xiaoyu.zhang5@student.rmit.edu.au; hai.dong@rmit.edu.au; zhifeng.bao@rmit.edu.au; xun.yi@rmit.edu.au
- *Shixun Huang is with the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia*
  Email: shixun_huang@uow.edu.au
- *Jiajun Liu is with CSIRO's Data61, Australia*
  Email: ryan.liu@data61.csiro.au
- *Xiaoyu Zhang and Shixun Huang have contributed equally*

Manuscript received 12 June, 2024; revised 30 September, 2024; accepted 16 October, 2024.

TABLE 1: Example of fine-grained workload in BS and EN

	s_1	b_3	b_4	b_5	s_2	b_6	b_7	b_8	b_9	s_3	b_1
t_1	1	4^*	0	1	4^*	0	0	3	1	0	0
t_2	0	3	1	0	3	1	1	4^*	3	0	1
t_3	4^*	0	1	0	2	1	0	3	4^*	1	2
t_4	3	2	2^*	2	0	2^*	2^*	1	1	2^*	3
t_5	2	1	0	3^*	1	0	2	0	3	1	4^*
C	13				16					6	
F	9				12					5	

[1] C, F represent coarse-grained and fine-grained workload metric respectively

corresponding resource allocation matter to the deployment cost under various user's delay tolerance values.

To the best of our knowledge, few researchers attempted to address the trade-off between cost and delay while considering the computation resource allocation [8, 19]. Despite that, existing studies are subject to the following major limitations. First, existing studies suffer from the *scalability* issue, making it impractical to deploy their solution over large-scale real-world datasets. In real-world cases, base stations are large in amount and the number is still increasing (e.g., Shanghai is projected to build 50 5G base stations per km^2 [20]). Designing a highly scalable and efficient solution is therefore essential (the detailed discussions can be found in Section 2). Second, the issue of delay has not been well addressed. Specifically, it has been ignored by most existing studies that the computation delay is supposed to decrease if more servers are placed in edge nodes.

It is worth noting that this is an extension of our previous short paper at ICSOC [21], which aims to address the trade-off between the deployment cost and delay under a coarse-grained scenario. In this paper, (1) inspired by our observation that the peak workload of each base station usually appears at different times, we formulate our problem to support the fine-grained case and accordingly propose a fine-grained optimization to further decrease the deployment cost. (2) We propose a series of solutions to improve the existing methods in terms of scalability and practice. All the above contributions are newly made in this work.

Overall, our **Main Contributions** include:

- We formulate a novel yet practical problem to address the trade-off between the deployment cost from service provider's perspective, and the transmission and computation delay from the user's perspective. We propose a peak workload metric-based measurement to guarantee the robustness of our deployment strategies; Moreover, we define a practical delay measurement to ensure its feasibility in real-world cases. (Section 3)
- We prove that our problem is NP-hard. (Section 4)
- We develop a suite of approaches to address this problem. Specifically, we propose a Cluster-based Mixed Integer Programming (MIP) method and a Distance Aware Coverage First Search (DA-CFS) algorithm to address the efficiency and effectiveness issue of MIP [8, 19] and Coverage First Search (CFS) [21], respectively. In addition, we propose a fine-grained optimization strategy to further improve the effectiveness of CFS and DA-CFS. (Section 5)
- We conduct comprehensive experiments over a large-scale real-world dataset to verify the performance of our proposed solutions. It turns out that our solutions have a great advantage in scalability compared with the baseline methods in both efficiency and effectiveness. (Section 6)

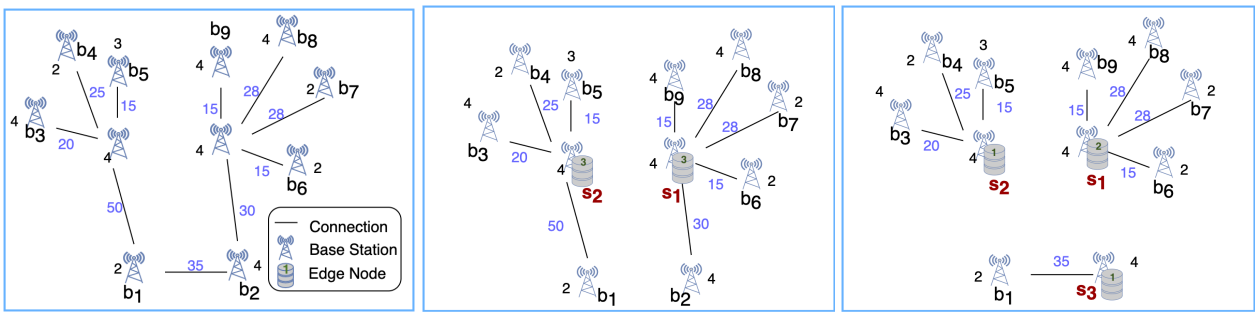
1.1 Example Use Case

Fig. 1 demonstrates how users' tolerance of service delay affects the optimal edge node deployment when considering cost-efficiency. In this figure, the blue numbers along with lines represent the distance while the black numbers with the based stations and edge nodes represent the workload. The workload and the distance together decide the transmission latency, while the workload matters to the computation delay. Basically, the larger workload would cause a larger computation delay and transmission delay, while the larger distance would further increase the transmission delay. Fig. 1a shows the initial connections between base stations which are candidate potential locations to deploy edge nodes (i.e., edge nodes co-locate with base stations). Our edge node deployment problem, given different delay thresholds, we want to find the corresponding optimal plan, which would not only guarantee that the workload in the base station can be processed within the given delay threshold but also the overall deployment cost can be minimized.

Developing an edge node within a base station will introduce a setup cost while adding servers to an edge node to increase its computing capacity will generate server purchase costs (the number along with edge nodes represent the number of server unit we deploy to each edge node). Given users' delay tolerance threshold and the goal of cost minimization, placing edge nodes at appropriate base stations with the right number of servers can significantly reduce the overall cost.

EN site selection with coarse-grained workload estimation. With the objective of minimizing the total cost, the optimal edge node deployment strategy would vary under different users' delay tolerance. Fig. 1b illustrates an optimal edge node placement in case the users' delay tolerance threshold is 22s, where the most cost-effective deployment is to develop two edge nodes S_1 and S_2, with each having 3 server units installed. In this case, developing one more edge node is more expensive than simply adding more server units to existing edge nodes. However, when we decrease the delay tolerance threshold to 16s, the optimal placement becomes what is shown in Fig. 1c. To satisfy this more rigorous delay tolerance requirement, there are two intuitive options: (1) continuously adding more servers to existing edge nodes to further decline the computation delay, or (2) installing a new edge node to decrease the transmission delay. Fig. 1c shows that the optimal solution is to install a new edge node instead of adding more servers.

Resource allocation with fine-grained workload estimation. Allocating appropriate resources to each edge node is another important factor that can directly affect the deployment cost. In Fig. 1c, the workload is measured in a coarse-grained manner by taking the peak workload of each location. The capacity required for s_1, s_2 and s_3 is 13, 16 and 6 respectively. Table 1 records the precise workload of each location in a series of time scales (i.e., t_1 to t_5). It can be seen that the peak workload (stared number) of each location appears at



(a) Initial network (b) OPT with 22s delay (c) OPT with 16s delay

Fig. 1: Example of optimal EN deployment under different delay tolerance

TABLE 2: Literature review of cost minimization problems

Ref	Server Setting		Delay			Workload Metric	# of BS	Method
	Capacity[1]	# in EN	Usage[2]	Transmission Delay	Computation Delay			
[19]	HO	≥ 1	O	✓	✗	AVG	20	MIP
[8]	HO	≥ 1	T	✓	✓	AVG	–	MIP
[9]	HE	1	T	✓	✗	AVG	10-50	Greedy
[7]	HE	1	T	✓	✗	AVG	200-1000	Greedy
[22]	HE	1	O	✓	✗	AVG	$\leq$20	Genetic
[10]	HE	1	T	✓	✗	AVG	100-500	Greedy
[23]	HE	≥ 1	T	✓	✓	AVG	¡3042	Heuristic
[24]	HE	≥ 1	T	✓	✓	AVG	$\leq$10	RL
Our	HE	≥ 1	T	✓	✓	Peak	200-3042	Cluster-based MIP, DA-CFS

[1] HO, HE represent homogeneous and heterogeneous server capacity respectively.
[2] O, T represent formulate delay as optimizing objective and threshold respectively.

different time. Let us see an example in the first big column in Table 1 for the workload estimation for edge node s_1 with its connected base stations b_3, b_4, and b_5. For the coarse-grained case, with peak-based workload estimation, we take the biggest workload from each edge node and base station (the stared number), which is 4+4+2+3=13. However, when we look into each timestamp, the peak actually appears at t_4, which is 3+2+2+2=9. It can be easily observed that the fine-grained workload is always smaller than the coarse-grained workload. Thus, a more accurate fine-grained workload estimation can decrease the deployment cost dramatically.

2 Related Work

Extensive studies have been carried out in the area of deploying cloudlets and edge nodes (also known as edge servers) to facilitate MEC in the past few years. The edge nodes (ENs) and cloudlets are very similar concepts, which are small-scale clouds with limited storage and computation capacity. They usually co-locate with cellular Base Stations (BSs) and WIFI Access Points (APs) [1]. Thus, we review the literature related to edge nodes and cloudlets together for the edge node deployment problem.

Most existing studies on edge node deployment either focus on minimizing the delay experienced by mobile users [2, 3, 4, 5, 6], or aim at minimizing the delay with the balanced workload among edge nodes [11, 12, 13, 14, 15, 16, 17]. However, the deployment cost, as an essential factor that limits the network QoS, has only been considered by a limited number of studies [7, 8, 9, 10, 19, 22]. As shown in Table 2, existing studies on cost-aware deployment can be divided into two categories. The first category focuses on placing flexible numbers of servers with homogeneous computing capacity in each edge node [8, 19]. It aims to minimize the cost of edge node setup and server purchase. The second category is about placing a minimum set of servers with heterogeneous computing capacity to edge nodes and only a single server is allowed for each edge node [7, 9, 10, 22].

Compared with the second category, the first category has higher flexibility in manipulating edge node capacity which depends on how many servers are placed. Server placing is also a part of the edge node deployment. The second category, however, is much simpler. It aims to pick one server with the most fitting capacity from a given set of servers with heterogeneous capacities. Thus, the first category has a much higher solution space than the second category.

Our work, as shown in Table 2, belongs to the first category. Qiang and Nirwan [19] focus on achieving a balance between cost and delay. They formulate the problem as a multi-objective optimizing problem. Sourav et al. [8] aims to find the optimal edge node deployment with minimized cost, given a user delay tolerance threshold. However, both of them have the following limitations: First, the scalability of the solutions is not verified (as both are based on MIP). They were only evaluated in small datasets, e.g., only 20 base stations used in [19]. Second, their delay models are less realistic. In [8], computation delay is ignored. Although the computation delay is considered in [19], it ignores the fact that allocating higher computation capacity to an edge node can shorten its computation time. To alleviate the aforementioned limitations, in this paper, we define a more practical delay measurement and propose an approximate solution with remarkable strengths in scalability.

It is worth noting that, to measure the workload, all

TABLE 3: Table of Notations

Notation	Description
s, S	Edge node; Set of edge nodes
b, B	Base station, Set of base stations
$s.n$	Number of servers placed in edge nodes
r, U	User request, Set of user requests
d	Distance
CT	Number of concurrent task
ξ	Single task size
μ	Single server computation rate
W	Workload
$\mathcal{A}$	Assignment
C_{trans}	Transmission capacity of the channel
C_{comp}	Computation capacity of the edge node
$\mathcal{B}$	Channel bandwidth
SP	Channel signal power
N	Channel noise
D	Delay
θ	Delay threshold

previous work adopts an average based-workload metric [7, 8, 9, 10, 19, 22, 23]. However, this measurement cannot have a robustness guarantee, as it may not be capable of handling rush hour cases. To address this issue, we propose a peak-based workload measurement, which precisely estimates the workload by considering the time.

Last, noting that many factors can be optimized for better network performance; for example, precisely scheduling tasks can further optimize the resource utilization [25, 26], and in the meantime, save energy cost for the end device [27]. Although such factors are non-negligible during network planning, similar to existing works in the field of edge node site selection as we reviewed, e.g. [7, 8, 9, 10, 19], these factors are not our focus. To be more specific, our goal is to optimize the network deployment at the very early planning stage by accurately selecting the edge node sites and the corresponding resource allocation, while the former targets for the post-planning stage.

3 Problem Formulation

In this section, we define the MEC network and its components. Also, we define the workload and delay measurement. Then, we formulate our problem with the goal of minimizing the deployment cost. For the frequently used notations, please refer to Table 3.

3.1 Preliminaries

MEC network. An MEC network consists of a set B of base stations (BSs) and a set S of edge nodes (ENs). Elements in both B and S are denoted by a tuple (id, lat, lng, n) where lat, lng and n represent latitude, longitude and the number of allocated servers respectively. By following existing convention [11, 13, 15, 17] that ENs are co-located with BSs, we claim that a base station is upgraded to an edge node if computation resources (i.e. servers) are added. By following a closely related work [8], we adopt the setting that multiple standard servers are allowed to be added into an EN to supply sufficient computation capacity. Then, we define our BS and EN formally as follows: (1) $\forall\ b \in B$, $b.n = 0$; (2) $\forall\ s \in S$, $s.n \geq 1$.

EN setup cost and server cost. We define two kinds of costs incurred during the deployment: *EN setup cost* and *server cost*. Let p_r denote the setup cost, which is the cost of upgrading a BS to an EN, such as infrastructure renting fee and construction fee. Let p_s denote the server cost, which is the cost of purchasing new servers for ENs. For example, installing a server to a base station will cost $p_r + p_s$, while adding a server to an edge node will simply cost p_s.

Connectivity and EN service range. Two BSs b_1 and b_2 are connected if they meet a certain delay threshold that is constrained by transmission delay and computation delay together. We define the service range of a EN s as all the connected BS, which is denoted as $R(s)$.

User Request. To complete a specific task, the mobile user would send a user request to a base station to request a stream of data. We define the user request r = (b_i, t_s, t_e), where b_i, t_s, t_e represent the base station that receives the request r, request start time and request end time, respectively. To access the computation resource in the network, mobile users send their requests either to an EN directly, or to a nearby BS which would further offload received requests to an EN for computation resources. In later sections, we use the terms *user request* and *task* interchangeably unless specified otherwise.

Assignment. Given that ENs may have their service range overlapped, the base stations may choose multiple surrounding ENs to offload user requests. However, since a user request needs continuous processing [28], each single user request cannot be processed by multiple ENs. We use the assignment to indicate whether the user requests are allowed to be offloaded to specific edge nodes from the designated base stations. The assignment relationship between user requests, base stations and edge nodes follows the criteria below: (1) One base station may offload its incoming user requests to multiple ENs. Each user request, however, can only be assigned to one EN for processing. In other words, once an EN takes a user request, it should hold the task until its end and should not migrate it to other ENs. (2) The selected ENs cover all BSs in the network, and hence are capable of handling all user requests.

We use $\mathcal{A}$ and $\mathcal{A}_t$ to denote the assignment at a coarse-grained base station level and a fine-grained user request level, respectively. For example, at the coarse-grained base station level, $\mathcal{A}(s_1) = \{b_1, b_2\}$ represents that base station b_1 and b_2 are assigned to edge node s_1, while at the fine-grained level, $\mathcal{A}_t(s_1) = \{r_1, r_2, r_3\}$ represents that at timestamp t, user request r_1, r_2, r_3 from their originated base stations are offloaded to s_1.

3.2 Workload Measurement

We propose a peak metric to measure the workload and introduce the concept of *Concurrent Tasks* to facilitate our peak workload measurement. We claim that two tasks are concurrent if they have their processing time overlapped. We use CT to denote the number of concurrent tasks.

A user request r will last a period of time, e.g. from $r.t_s$ to $r.t_e$, for a specific task. We assume that users are all requesting data-intensive computing tasks, e.g. HD videos, and each r requests ξ of data at any timestamp. We make this assumption to simulate an extreme case with which we can guarantee that our deployed MEC network is capable of dealing with an overwhelming workload.

There are two kinds of workload during a time period, e.g. from t_1 to t_2. We mainly focus on: (1) the peak transmission workload between b_i and edge node s_j (Equation. 1); (2) the peak computation workload in an edge node s_j (Equation. 2). We define those in the following equations:

$$W_{t_1,t_2}(b_i, s_j) = \xi \cdot \max\{CT_{trans}(t, b_i, s_j) : t \in [t_1, t_2]\} \quad (1)$$

$$W_{t_1,t_2}(s_j) = \xi \cdot \max\{CT_{comp}(t, s_j) : t \in [t_1, t_2]\} \quad (2)$$

where $\max\{CT(t,\cdot) : t \in [t_1, t_2]\}$ is to find the maximum number of concurrent tasks appearing between t_1 and t_2.

We can perform a coarse-grained workload estimation for an edge node s at the base station level. We suppose that for a $b \in \mathcal{A}(s)$, the largest count of concurrent tasks appearing at a time point (timestamp t_i), is $CT_{t_i}(b)$. The corresponding workload of b can then be estimated as $W_{peak}(b) = \xi \cdot CT_{t_i}(b)$. Thus, the workload of b at any timestamp $t' \neq t_i$ should not have the workload exceed the peak, i.e., $W_{t'}(b) \leq W_{peak}(b)$. If we ignore the case that for each connected base station, the appearance time of peak workload is varying, we can do a coarse-grained workload estimation for s as $W(s) = \sum_{b_i \in \mathcal{A}(s)} W_{peak}(b_i)$.

3.3 Delay Measurement

There are two major types of delays to be addressed: the *transmission delay* for the delay of transmitting a user request between a BS and an EN, and the *computation delay* for the delay of processing a user request in an EN [29]. Those two delay types are related to the *channel's transmission capacity* and *EN's computation capacity* respectively.

Transmission capacity. We adopt Shannon's channel capacity formula [28] to compute channels' transmission capacity:

$$C_{trans} = \mathcal{B} \log_2 (1 + \frac{SP}{N}) \quad (3)$$

In this equation, $\mathcal{B}$ represents the channel's bandwidth, SP represents the average received signal power over the channel and N represents the average noise power over the channel. We assume that the signal power is identical to all channels. Considering that channel noise can be affected by many factors, such as distance, environments and quality of cable [30], we use a very common way in the literature by assuming that the channel noise is only affected by the distance [5, 11, 13, 15]. We define the noise as $N = \alpha \cdot d(s, b)$, where $d(s, b)$ denotes the distance between s and b, and α is a coefficient between N and the distance.

Computation capacity. Adding servers to an EN grants it more computation capacity as discussed in [8]. We assume that servers placed to EN have the same computation capacity μ bit/s. Then, for an EN s with $s.n$ servers placed, the computation capacity is:

$$C_{comp} = s.n \cdot \mu \quad (4)$$

Delay calculation. The delay calculation relies on the data size and processing capacity [31]. By referring to our previously defined peak workload metric, we estimate the delay for processing a user request r according to the maximum workload in both transmission between BS b_i and EN s_j and computation in s_j, which is modeled as follow:

$$D(r) = \frac{W_{r.t_s, r.t_e}(r.b_i, s_j)}{C_{trans}} + \frac{W_{r.t_s, r.t_e}(s_j)}{C_{comp}} \quad (5)$$

where the first term represents the transmission delay for task r from b_i to s_j, while the second term represents the computation delay experienced by r in edge node s_i.

Similarly, we can also perform a coarse-grained estimation of delay at the base station level by referring to the coarse-grained workload measurement in Section 3.2. The maximum delay incurred between a BS b and an EN s can be modeled as:

$$D(b, s) = \frac{W_{peak}(b)}{C_{trans}} + \frac{W_{peak}(s)}{C_{comp}} \quad (6)$$

3.4 Problem Definition

To make our problem definition clean and intuitive, we first give the definition of the Qualified EN Placement Plan, which precisely defines the constraints needed for a qualified placement plan (as shown in Definition 1). Furthermore, we pick the optimal placement plan from the qualified EN placements with the goal of minimizing the cost, which is defined in Definition 2.

Definition 1 (Qualified EN Placement Plan). Given a set B of BSs in which each has a set of user requests attached and a delay threshold θ, select a subset $S \subseteq B$ as ENs such that the following constraints hold:

(1) For each single user request r from a $b \in B$, the delay experienced is always within the given threshold θ. i.e. $\forall\ r \in b.U\ b \in B$, $D(r) \leq \theta$ (see Equation 5 for the delay calculation);

(2) One base station may have multiple designated ENs to offload tasks, which means user requests from one BS may be assigned to different ENs. However, one user request can only be processed by one EN at a time t. i.e., $\forall s_i, s_j \in S, s_i \neq s_j$, $\mathcal{A}_t(s_i) \cap \mathcal{A}_t(s_j) = \emptyset$;

(3) Once a user request r has been offloaded to an EN s at time t_k, e.g. $r \in \mathcal{A}_{t_k}(s_i)$, it will not be assigned to another EN later. That is, r is either completed at time t_k, i.e., $r \notin \bigcup_{s \in S} \mathcal{A}_{t_{k+1}}(s)$, or being continually processed by the same EN at later timestamps, i.e., $r \in \mathcal{A}_{t_{k+1}}(s_i)$.

Definition 2 (Cost Minimization in MEC Edge Node Placement). The **CMMENP** problem is to find an optimal EN placement plan S^* which can minimise the total deployment cost, i.e.,

$$F(S^*) = \arg\min_{S \subseteq B} \sum_{s \in S} (p_r + s.n \cdot p_s) \quad (7)$$

where $F(S^*)$ denotes the total cost incurred by selecting S^* as ENs, S is a qualified EN placement plan, p_r is the setup cost, and p_s is the server cost.

4 Hardness Analysis

In this section, we prove the NP-hardness of our problem through a reduction from a NP-hard problem, namely the Dominating Set (DS) problem [32].

Definition 3 (Dominating Set Problem). Given a graph $G = (V, E)$, where V, and E are the vertex set and the edge set respectively, we aim to find the smallest dominating set $D^* \subseteq V$ such that each $v \in V \backslash D^*$ is adjacent to at least one item in D^*.

Theorem 1. CMMENP is NP-hard.

Proof: We prove that CMMENP is NP-hard through a reduction from the Dominating Set (DS) problem. The reduction is performed in three main steps:

(1) Let us consider a special case of CMMENP in which the server cost $p_s = 0$ and the construction cost $p_r = 1$. In such case, an edge node can have unlimited computation capacity without incurring any server placement cost. Therefore, the total cost only depends on the number of edge nodes being placed and the computation delay can be ignored by adding a very large number of servers to edge servers. In this case, θ is related to transmission delay only.

(2) We construct a graph for our CMMENP problem $G = (B, E')$, where B is the set of base stations and E' is the set of connections/edges between base stations in B. For any $u, v \in B$, there is an edge (u, v) between them, if and only of the transmission delay between u and v is not greater than θ. Since we set $p_s = 0$, the effect of transmission delay upon computation delay can be ignored such that we can ignore the edge weight as well. Thus, the input graph is an indirect unweighted graph, which is identical to the input of DS. Therefore, for any instance of DS problem with V and E, we can construct an instance of the CMMENP problem by mapping V and E in DS to the base station set B and connections E' in CMMENP, respectively.

(3) The objective of our CMMENP problem is to minimize the cost. Given p_r=1 and $p_s = 0$, it is equivalent to finding the smallest subset $S^* \subseteq B$ that can cover all $b \in B \backslash S^*$, where S^* is also the optimal solution D^* to the DS instance. If there exists a polynomial time algorithm which can find S^* for the CMMENP instance, the DS problem can also be solved in polynomial time, which is not possible unless P=NP. Thus, our CMMENP problem is NP-hard.

□

5 SOLUTIONS

Our solutions can be categorized into two streams: Mixed Integer Programming (MIP)-based methods (in Section 5.1) and heuristic methods (in section 5.2). The first stream exhibits its advantage in effectiveness, while the heuristic stream solutions have a great advantage in efficiency.

The first stream exhibits its advantage in effectiveness; however, those methods are not scalable to large-scale data due to the low efficiency. To be more specific, with the MIP-based methods, we formulate CMMENP as a linear optimization problem by following [21]. However, due to the fact that this method is not scalable to large datasets, we propose a cluster-based MIP to improve its efficiency. On the other hand, while the cluster-based MIP can improve efficiency to a large extent and still gains benefits from the unsurpassable effectiveness of the MIP, it faces a trade-off between efficiency and effectiveness brought by the cluster size (please refer to our experiments in Section 6.2.1).

The heuristic stream solutions have a great advantage in efficiency, e.g. Coverage First Search (CFS) [21] and Distance Aware Coverage First Search (DA-CFS) in Section 5.2.1 can solve the problem in cubic time. To further utilize their high efficiency and to mitigate their disadvantage in effectiveness, based on our fine-grained workload and delay measurement as defined in Section 3.2 and Section 3.3, we propose a fine-grained optimization in Section 5.2.2. With the fine-grained resource allocation, we further decrease the deployment cost which can even outperform MIP-based solutions in terms of effectiveness.

Algorithm 1: Cluster-Based MIP

Input : Base Station set B, Minimum Cluster Size β, User Requests U
Output: Cluster set C

```
C ← ∅;
while |B| > β do
    Build hierarchy tree H of B;
    // cut tree at various height
    foreach h ∈ H do
        C' ← cut H at height h;
        flag ← false;
        foreach c' ∈ C' do
            if |c'| ≥ β then
                flag ← true;
                C ← C ∩ c';
                B ← B\c';
                break;
        if flag then
            break;
C ← C ∩ B;
S ← ∅;
foreach r ∈ U do
    Calculate workload W and delay D by U;
    S' = MIP(W, D);
S ← S ∩ S';
return S;
```

5.1 Cluster-based MIP

In this stream, we utilize the MIP solution as our baseline by following the formulation in [21].

Since the running time of MIP increases exponentially with the growing size of the input set, we propose a cluster-based MIP method to alleviate the scalability problem of MIP while taking advantage of its high effectiveness. We have the following three goals when partitioning the input base stations into clusters: (1) The cluster should be independent of each other. In other words, each cluster should have a clear border that separates points inside and outside the cluster. (2) The cluster should be balanced, which means the size of each cluster should not differ too much. We will never expect a very large or tiny cluster, as a large cluster will also face the efficiency issue with MIP while small clusters are potentially degrading the effectiveness. (3) The partition should be achieved efficiently. Such a partition can be achieved as a k-cut problem in graph theory, which is known as NP-hard [33]. Thus, it is necessary to find an approximate solution that can solve the problem efficiently.

To achieve the first goal, we adopt hierarchical agglomerative clustering that forms clusters by gradually merging the clusters nearby. This merging mechanism guarantees that the formed clusters are independent and can avoid points that span across clusters to the largest extent. However,

when it comes to the second goal, the original hierarchical agglomerative clustering algorithm has no guarantee of the balance between clusters, especially when the target dataset has one single center like our case (please refer to the data distribution in Fig. 7). Thus, we further modify the hierarchical clustering to generate relatively balanced clusters, whose details are presented in Algorithm 1.

Specifically, we first adopt a traditional way [34] to build a hierarchy tree of cluster/data points (line 3). By cutting the dendrogram at various heights, we can obtain different agglomerative formations for clusters. Since we do not expect to have extremely large or small clusters, we cannot cut the tree directly to obtain clusters, as this would return extremely unbalanced results in most of the cases. Instead, we iteratively search each level of the tree in a bottom-up manner to find a cluster that is larger than the threshold we set (lines 8 to 12). Once we find a cluster that meets the threshold, we remove this cluster of base stations from the base station set (line 11). Next, we rebuild the hierarchy tree with the rest of the base stations and iterate the above process to find the next cluster until no enough base stations being left (line 13). The remaining base stations then form the last cluster (line 15).

In terms of the last goal of efficiency, we adopt an efficient hierarchical agglomerative clustering algorithm with the time complexity $\mathcal{O}(n\log^2 n)$ [34] for our hierarchy tree construction. The time spent searching for a cluster with an appropriate size is $\mathcal{O}(n\log n)$. Thus, the overall time complexity of our clustering algorithm is $\mathcal{O}(n^2\log^3 n)$.

After we partition base stations into different clusters, we apply MIP in each cluster, as shown in Algorithm 1 (lines 17 to 19). As a result of the reduced searching space, MIP is capable of finding a sub-optimal solution in an acceptable time period for each cluster locally. Then, we gather those local solutions as the overall result (line 20). Although the cluster-based MIP method slightly sacrifices the accuracy of MIP, it increases the scalability dramatically.

5.2 Heuristic-based Methods

In this section, we take the Coverage First Search (CFS) method in [21] as our baseline. Then, we improve its effectiveness by proposing a Distance Aware Coverage Search (DA-CFS) method. Also, we introduce a fine-grained optimization solution to further decrease the deployment cost.

5.2.1 Distance Aware Coverage First Search (DA-CFS)

As discussed in [21], although CFS can solve the problem in polynomial time, its effectiveness is very limited. According to our observation, the CFS has the following limitations: (1) EN candidates are limited. A BS that has been assigned will no longer be considered as an EN candidate. However, those assigned BSs are potentially suitable EN candidates, as they might well cover their neighboring BSs. (2) The assignment between BSs and ENs has not been well addressed. Assigning a BS to a distant EN will cause high transmission delay, which decreases the computation delay tolerance, so excessive EN capacity is required. Thus, it is always preferred to assign a BS to a nearby EN considering cost minimization.

Hence, we propose a Distance-Aware Coverage First Search (DA-CFS) method (Algorithm 2) to address those limitations. In DA-CFS, we define a candidate set, which collects BSs that are assigned to ENs but are potentially competitive EN candidates. Once a new EN is selected, the BSs that are distant from it is fed into the candidate set. We define a threshold τ as the maximum number of BSs selected into the candidate set from each EN's selection. We also introduce an inverted index $\mathcal{M}$ of $\mathcal{A}$, which maintains a set of key-value pairs to record the assignment between each pair of BS and EN.

Algorithm 2: DA-CFS Algorithm

Input : Base Station set B, Delay threshold θ, Candidate number τ
Output: Edge Node set S

1 $S \leftarrow \emptyset$, $C \leftarrow \emptyset$, $\mathcal{M} \leftarrow \emptyset$, $\mathcal{A} \leftarrow \emptyset$; // C: candidate set, $\mathcal{M}$: inverted index of $\mathcal{A}$, a set of $\langle b : s\rangle$
2 **while** $B \neq \emptyset$ **do**
3 $B \leftarrow$ getConnection(B, θ);
4 $b_s \leftarrow \arg\max\{|R(b)| \mid b \in \{B, C\}\}$; $S \leftarrow S \cap b_s$;
5 **if** $\mathcal{M}[b_s] \neq \emptyset$ **then**
6 s $\leftarrow \mathcal{M}[b_s]$, $\mathcal{M}$.remove(b_s);
7 **foreach** $b \in R(b_s)$ **do**
8 **if** *distance(b, b_s) <distance(b, s)* **then**
9 $\mathcal{M}[b] \leftarrow b_s$; $\mathcal{A}[b_s]$.add(b);
10 $\mathcal{A}[s]$.remove(b);
11 **else**
12 $\mathcal{M}[b] \leftarrow b_s$;
13 Sort $\mathcal{A}[b_s]$ by distance in a descending order;
14 Add the first τ BSs from $\mathcal{A}[b_s]$ to C;
15 Remove b_s from B or C;
16 Remove all the items in $\mathcal{A}[b_s]$ from B;
17 **return** S;

In Algorithm 2, if a BS in the candidate set connects to more unassigned BSs compared with other BSs in the input BS set B, we will select it as an EN; otherwise, we will select an EN from the set B by following the CFS method (lines 3-4). If a new EN is from the candidate set, we will first disconnect it from its previously assigned EN (lines 6). As some BSs might locate in more than one ENs' service area, we will re-assign those BSs to the new EN if their distances to it are shorter (lines 7-10). All the assigned BSs will be registered in $\mathcal{M}$ (line 12). For each new EN, we will add its τ most distant BSs into the candidate set (lines 13-14). We conduct experiments to obtain an appropriate value of τ in Section 6. Finally, we remove the selected EN and its assigned BSs from the input set B (lines 15-16).

Time Complexity Analysis. The primary computation cost of the DA-CFS method still arises from the BS connection computing ($\mathcal{O}(n^2)$) and the sorting process ($\mathcal{O}(n\log n)$). Thus, its overall time complexity is the same as the CFS method, which is $\mathcal{O}(n(n^2 + n\log n))$.

5.2.2 Fine-grained Optimization

To further optimize the cost generated by CFS and DA-CFS, we introduce a fine-grained optimization solution to those two methods for a more precise resource allocation.

We observe that our proposed peak workload metric (in Section 3.2) inevitably produces higher deployment costs.

We also find that the wildly adopted average workload metric has no guarantee of the robustness of the network. Under the average workload metric, the deployed EN may not have enough computation resources to process all user requests during rush hours. According to our observation, in a city, the peak workload is more possibly to appear at working hours for base stations located in the CBD area, while it is more likely to appear in the residential areas at off-work hours. Based on the above observation, we perform a fine-grained re-assignment at the user request level to delicately identify how much computation resources are required for each EN and thus further decrease the deployment cost.

We propose a fine-grained estimation of the real capacity required for the EN as shown in Algorithm 3. Instead of computing an exact result by looking into each of the user requests, we perform an estimation for the result, which will allocate a bit more computation resource to each EN. This estimation results from the following two reasons: (1) Due to the large volume of user requests in the network, it is time-consuming to perform computation based on each of the user requests. (2) We expect that our deployed network can support future development where more users and data-intensive apps may join this network. We control the cost in a range that provides a robustness guarantee for the present and foreseeable future.

Algorithm 3 demonstrates our fine-grained EN capacity estimation. First, We reassign each user request to its nearest edge node (line 1). Second, we perform a fine-grained resource allocation to EN. As explained before, we work on an estimated value rather than an exact value. In our fine-grained resource allocation, we perform the estimation based on two kinds of workload: the computation workload in EN and the transmission workload between a BS and an EN. We estimate the computation required of EN via the following procedures: (1) To estimate the computation workload in EN, we iteratively compute the peak workload of each ENs by looking into each timestamp, including the request start time and end time, of the user request (lines 3 to 6). (2) To estimate the transmission workload, we further look into user requests in each assigned base station, e.g., $b \in \mathcal{A}(s)$ (lines 7 to 16). For each b, we find all the user requests that are offloaded to the assigned EN (line 9). Then, we calculate the maximum workload during the transmission (lines 10 to 12). (3) With both the computation workload and the transmission workload, we can estimate the computation capacity required for s (line 13). Eventually, we take the maximum value as our estimated capacity for s (line 16). By optimizing with our fine-grained EN capacity estimation and the corresponding resource allocation, the cost generated decreases dramatically, which can be seen from our experiment analysis in Section 6.

Algorithm 3: Fine-grained EN Capacity Estimation

Input : Edge Node set S, User Requests U
Output: Edge Node set S with capacity

```
// 1: assign user requests
1  Assign each r ∈ U to its nearest EN;
// 2: Fine-grained resource allocation
2  foreach s ∈ S do
3      Get all timestamps T from user requests to s;
4      Sort T in asc order;
5      t_s ← T[0]; t_e ← T[|T| − 1];
6      w_s ← W_{t_s,t_e}(b_i, s_j);
       // Get the max required capacity c
7      c ← 0;
8      foreach b ∈ A(s) do
9          Get all timestamps T' from user requests to b;
10         Sort T' in asc order;
11         t'_s ← T'[0]; t'_e ← T'[|T'| − 1];
12         w_b ← W_{t'_s,t'_e}(b);
13         c' ← getCapacity(w_s, w_b, dist(b, s));
14         if c' > c then
15             c ← c';
16     s.capacity ← c;
17 return S;
```

6 Experiment

In this section, we perform a set of experiments to explore the following research questions:

RQ1. What is the efficiency, effectiveness and scalability of the proposed methods? (Section 6.2.1, Section 6.2.2 and Section 6.2.3) What are the reasons behind the performance? (Section 6.2.4)

RQ2. How does the value of delay threshold θ affect the result (see Section 6.2.5)?

RQ3. What is the performance respectively with the average workload and the peak workload? (Section 6.2.6)

RQ4. What is the performance of the proposed clustering method (Algorithm 1) in terms of the three goals depicted in Section 5.1? (Section 6.2.7)

RQ5. τ as maximum number of BSs selected into a candidate set of an EN for DA-CFS, what is its optimal value? (Section 6.2.8)

6.1 Experiment Settings

6.1.1 Dataset

Our experiments are conducted on the Shanghai Telecom Dataset[1], which contains the user service records of 3042 base stations in Shanghai [11, 15, 35]. The experimental dataset contains 1,048,576 valid user service requests received by these base stations during a time period of 6 months. To support our proposed peak workload metric, we conduct statistical analysis on the concurrent tasks. It is found that the maximum number of concurrent tasks received by a base station ranges from 0 to 23, which shows high workload diversity.

6.1.2 Experiment Environment

All the experiments are conducted on a computer with Linux system, Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz and 512GB memory. The MIP approach is implemented using the CPLEX[2] library. All the other methods are implemented in Java. Codes are publicly available at[3].

1. Dataset: http://sguangwang.com/TelecomDataset.html
2. CPLEX: https://www.ibm.com/au-en/analytics/cplex-optimizer
3. https://github.com/PetalZh/MECExpJava

TABLE 4: Parameter Setting

Parameter	Value
Standard Server Computing Capacity μ	100 bps
Channel Bandwidth B	5 Hz
Channel Signal Power SP	-35 dBm
Single Task Size ξ	15 Mb
Default Delay Threshold θ	22.0 s
EN Construction Cost p_r	400
Single Standard Server p_s	100

TABLE 5: Estimated EN coverage with different θ value

θ	14.0	16.0	18.0	20.0	22.0	24.0	26.0
Coverage (m)	396	618	880	1175	1497	1841	2204

6.1.3 Methods for Comparison

We compare the performance of the following methods:

- **MIP** [8, 19] – It solves the problem as a integer programming problem. We set its maximum running time to 1 hour in each run, considering the extremely high time consumption of MIP on large datasets.
- **CFS** [21] – A heuristic solution that greedily selects a BS with the maximum coverage as EN.
- **Cluster-based MIP (MIP+Cluster)** – It improves the efficiency of MIP while maintaining its high effectiveness.
- **DA-CFS** – A CFS with improved effectiveness by introducing a candidate list.
- **Fine-grained optimization for CFS (CFS-F) and DA-CFS (DA-CFS-F)** – Improved CFS and DA-CFS by peforming user-request-level resource allocation optimization.

6.1.4 Parameter Settings

Our parameter settings are summarised in Table 4. By following [8], we set the ratio between the construction cost of an edge node and the installation cost of a standard server to 4:1, and the computing capacity of a standard server to μ = 100 bps. The bandwidth $\mathcal{B}$ is set to 5 Hz and the channel signal power SP is set to -35 dBm[4]. The single task size ξ is configured to 15 Mb[5]. For the delay threshold θ, we set its value from 14.0s to 26.0s by referencing users' average delay tolerance on mobile applications [36]. With the settings above, we estimate the coverage of an EN on different delay thresholds, which is shown in Table 5. Furthermore, all our experiments are conducted with a default delay threshold θ = 22s based on the empirical studies depicted in Section 6.2.5.

6.1.5 Evaluation Metrics

The effectiveness metrics include the deployment cost and the number of selected ENs, while the efficiency is assessed by the running time (with the average time of 5 runs).

We measure the effectiveness and efficiency of all the candidate solutions on different BS input scale to verify their scalability. We also evaluate the effectiveness and efficiency of these candidate solutions with different delay threshold θ to investigate how the dela y threshold value impacts their performance. An exclusive scalability analysis is conducted in terms of the effectiveness and efficiency.

4. https://www.securedgenetworks.com/blog/wifi-signal-strength
5. https://www.reviews.org/au/internet/data-usage/

6.2 Experimental Results

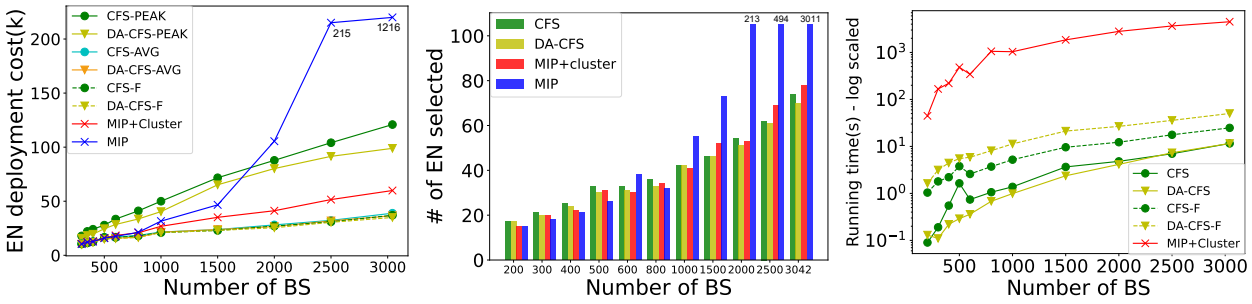
6.2.1 Effectiveness Analysis

First, we compare the effectiveness of these solutions based on the deployment cost. As shown in Fig. 2a, MIP shows a remarkable cost-saving advantage when the input BS scale is small (when input BS scale $\leq$ 1500). However, MIP is unscalable to larger input BS sets so that it fails to produce high-quality solutions with the time limit (i.e., one hour). In contrast, CFS generates a higher cost compared with MIP in small BS input scales. However, since CFS is an efficient algorithm, the problem can always be solved in a shorter time period. From the figure, we can see that CFS can easily beat MIP on BS inputs with larger scales (when input BS scale $\geq$ 2000).

As explained in Section 5, our proposed cluster-based MIP (MIP+Cluster) and DA-CFS focus on improving MIP and CFS in terms of efficiency and effectiveness respectively. In Fig. 2a, we can observe that MIP+Cluster has a very similar performance with MIP when the size of the BS input is less than 1000. With the further increase of the BS input scale, MIP+Cluster maintains a steady rising trend with the deployment cost staying at relatively low values. Thus, we can claim that our strategy of breaking the input BS set into small clusters and finding local solutions in each cluster is an effective way to address the efficiency issue of MIP.

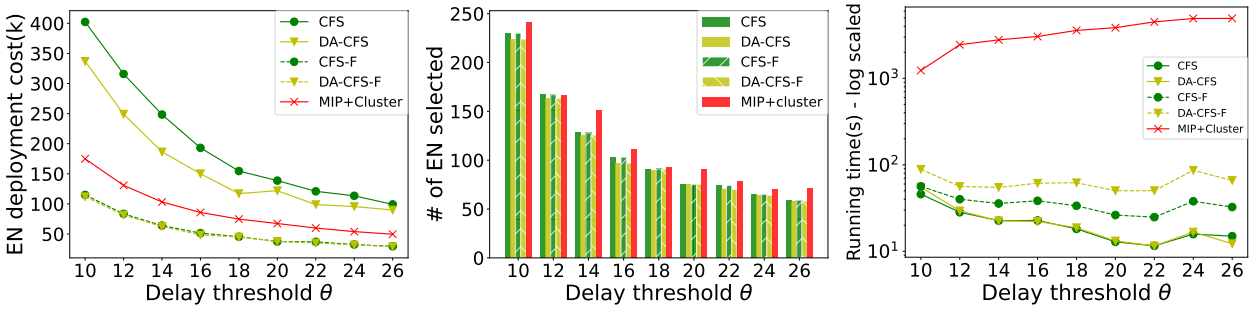
DA-CFS aims at improving the effectiveness of CFS. CFS is incapable of finding all potentially suitable EN locations and optimizing the assignment between BSs and ENs. This causes the problem that its selected ENs may not be in the optimal locations and the ENs would require high computation capacity to serve distant BSs. DA-CFS solves this issue with a candidate list and assignment optimization. From Fig. 2a, we can find that, DA-CFS always outperforms CFS in cost-saving. This advantage becomes increasingly obvious when the number of participated BSs rises. However, the cost produced by DA-CFS is still relatively high compared with MIP+Cluster.

Second, we investigate the number of ENs selected by each method to analyze the reason behind the high cost of CFS and DA-CFS, as shown in Fig. 2b. We can see that DA-CFS always selects fewer EN locations compared with CFS and thus generates lower costs. This matches the design incentive of DA-CFS that aims to select fewer ENs. We can also observe that the number of ENs selected by MIP+Cluster is not the lowest and sometimes higher than CFS and DA-CFS. Considering that MIP+Cluster incurs comparatively low cost in total, we can conclude that decreasing the number of selected ENs is not the only way to minimize the cost. An elegant and precise assignment also plays an important role (a further discussion on the quality of the assignment can be found in Section 6.2.4). Therefore, it inspires us to improve CFS and DA-CFS by elegantly optimizing the assignment to further decrease the cost. That is why fine-grained optimization is essential to CFS and DA-CFS with the goal of cost-saving. As we can see from Fig. 2a, by applying our fine-grain algorithm (Algorithm. 3) to precisely address the



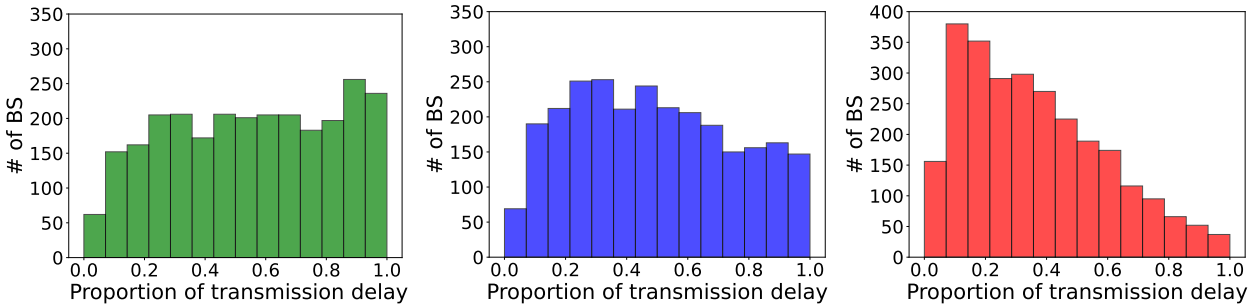
(a) Deployment cost (b) # of EN selected (c) Running time

Fig. 2: Effectiveness and efficiency with different BS input scale



(a) Deployment cost (b) # of EN selected (c) Running time

Fig. 3: Effectiveness and efficiency with different θ value



(a) CFS (b) DA-CFS (c) MIP+Cluster

Fig. 4: Distribution of BSs with different transmission delay proportions on θ

assignment and the resource allocation, the costs of CFS and DA-CFS drop remarkably, which are even lower than MIP+Cluster.

6.2.2 *Efficiency Analysis*

Fig. 2c shows the running time of those methods in log scale. MIP+Cluster shows the highest running time (according to our experiment data, it requires about 90 minutes for the full dataset). In contrast, the running time for CFS and DA-CFS is significantly shorter, which is only around 10 seconds. Since CFS and DA-CFS have the same time complexity, the running time for these two methods is quite close. We also find that the running efficiency highly relies on the speed of draining the base stations in the dataset. In other words, the less EN selected, the faster the algorithm. For the CFS, we can observe an abnormal increase when the size of BS input is 500. From Fig. 2b we can also find the number of

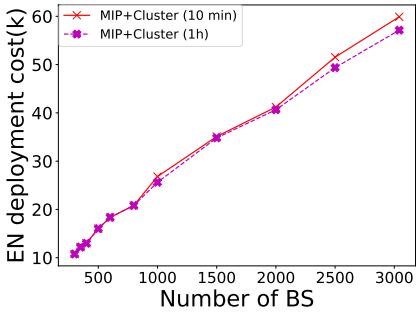


Fig. 5: Cost generated by MIP+Cluster with running time limit of 10min and 1h per cluster

ENs selected by CFS is also a relatively large value when the number of BSs is 500. Thus, it verifies our analysis that

the actual running speed of the algorithm depends on the BS coverage speed, to a large extent. Also, by combining Fig. 2b and Fig. 2c, we can see that DA-CFS is more stable than CFS in terms of the EN selection.

It is noteworthy that we exclude MIP in the previous comparison since we have to terminate it when the time limit (i.e., one hour) is reached. Furthermore, we set the time limit of 10 minutes for running MIP in each cluster. We also conduct an experiment to verify whether the longer running time can produce a better result for MIP+Cluster. As demonstrated in Fig. 5, we set the time limit of MIP in each cluster to 10 minutes and 1 hour respectively. Even though the problem in each cluster cannot be fully solved in 10 minutes, running one hour does not contribute too much on saving the cost. Thus, considering that the design objective of MIP+Cluster is to improve the efficiency, we adopt 10 minutes running time limit for each cluster.

We optimize CFS and DA-CFS by conducting a fine-grained resource allocation. Since it is implemented in the user request level that needs to deal with a large volume of data, the running time becomes inevitably longer. However, the running time is still in an acceptable range, as the problem can be solved within 30s and 1 min for CFS-F and DA-CFS-F respectively.

6.2.3 Discussion on Scalability

Recall our discussion in Section 6.2.1 and Section 6.2.2, MIP+Cluster shows a great advantage in cost-saving among all the proposed methods due to its outstanding performance in EN selection and BS assignment. Although the cluster-based method reduces the running time to a large extent, it still requires relatively long running time. If we further increase the size of the BS input set, the running time will be inevitably further boosted and the effectiveness will also be compromised, since more clusters will be introduced. In contrast, our DA-CFS has extraordinary efficiency. Even though it performs moderately in terms of the cost-saving objective, the overall trend shows that the performance does not degrade when the BS input scales up. Furthermore, because of its fast processing speed, the fine-grained optimization can be integrated, which addresses the effectiveness issue without sacrificing too much efficiency. Overall, DA-CFS-F shows better scalability than MIP+Cluster in both the effectiveness and the efficiency.

6.2.4 Transmission and Computation Delay in θ

Fig. 4 illustrates the distribution of BSs with different transmission delay ratios (to the total delay) for CFS, DA-CFS, and MIP+Cluster. The experiments are conducted on the whole BS set (i.e. 3,042 base stations) to investigate how our proposed methods reach the goal of cost minimization in terms of the quality of assignment conducted by each of the proposed methods. In those graphs, we evenly divide the transmission delay ratio in θ from 0 to 1 into 15 bins. The higher the bar, the more BSs fall into this specific bin. For example, in Fig 4a, the number of BSs that fall into the transmission delay proportions of 0 to 0.2 is 75+190+210=475.

To achieve the goal of cost minimization, a BS is always expected to be assigned to its nearest EN. As a result, the transmission delay can be lower and have more space left for the computation delay. Consequently, the computation resource required from the assigned EN is decreased.

By observing the distribution illustrated in Fig. 4, we can analyze how the assignment affects the cost generated. As shown in the figure, CFS generates a high transmission delay for a large proportion of BSs. Compared with CFS, we can see that the overall distribution of the BSs in DA-CFS shifts left. It verifies that the candidate list in DA-CFS and the corresponding BS assignment adjust strategy can decrease the overall proportion of transmission delay. It also explains why DA-CFS can always generate a lower cost than CFS. As aforementioned, MIP+Cluster has a great advantage in elegantly addressing BS assignments. From Fig. 4c, we can observe that most of the BSs fall into lower transmission proportions, which means most of the BSs generate lower transmission delay and θ is mostly taken by computation delay.

From all the observations above, it can be concluded that both EN selection and BS assignment play important roles in achieving the objective of the deployment cost minimization.

6.2.5 The Impact of θ

We evaluate the effectiveness and efficiency of the four candidate solutions by varying θ from 14s to 26s [36]. We can roughly estimate a value as the EN coverage range (radius) for reference by taking θ completely as the transmission delay. Considering the parameters we set for the channel transmission capacity, including bandwidth and noise-distance coefficient, and taking the largest workload in the network, we can calculate an estimated coverage range. For the θ value we take, the coverage spans from 396m to 2204m (see Table 5). Specifically, we study how θ affects the deployment cost, EN selection, and running time by conducting our experiment on all 3042 base stations.

It can be seen from Fig. 3a that higher θ values generate lower costs. This is because a loose delay constraint expands the service area of an EN so that the number of required ENs decreases (see Fig. 3b). However, as observed in Fig. 3a, the effect of θ on the deployment cost becomes weak when it reaches 22.0. That is because the number of selected ENs is relatively stable but the number of servers required by the ENs is increasing to cover their rising BS assignment. Therefore, we set θ to 22.0s in our experiments. In addition, it can be seen that the relative positions among the candidate solutions are unchanged along with the increasing threshold value for both the deployment cost and the running time. This indicates the weak impact of the threshold value on their relative performance comparison.

6.2.6 Peak vs. AVG

As mentioned in Section. 5.2.2, the peak workload metric may generate extra cost in the coarse-grained case compared with the average workload metric. However, from Fig. 6, we can observe that the costs generated by the peak workload metric are even smaller than the average workload metric by precisely considering the offloading time of each user request (i.e., the fine-grained methods). Also, the peak workload metric is more robust to dynamic variation of the workload in each server. We do experiment to verify the rate of violation on fulfilling all users' latency requirement

TABLE 6: Cluster results on different BS input scale

BS #	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Time (s)
1500	**317**	315	228	222	245	172	–	–	–	–	6.21
2000	248	291	297	228	**370**	306	259	–	–	–	13.08
2500	247	352	288	341	321	264	300	**386**	–	–	20.22
3042	331	327	303	**336**	247	256	327	329	245	340	54.43

TABLE 7: Cost and # of EN selected with different τ in DA-CFS

τ	6	8	10	12	14	16	18	20	22	24	26
EN #	164	164	163	162	163	162	162	162	162	162	162
Cost(k)	362.0	343.4	360.5	358.4	357.2	354.1	353.1	346.8	235.9	232.7	230.5

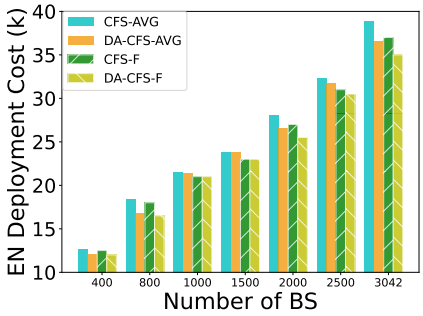


Fig. 6: Deployment cost: Peak vs. AVG

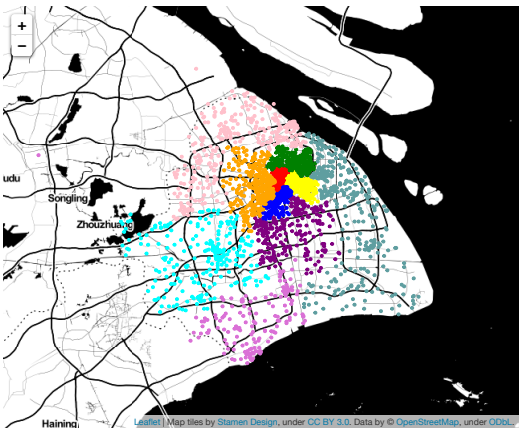


Fig. 7: Visualise clusters on Shanghai Telecom Dataset (the color indicates different clusters)

for both average and peak workload metrics. According to our result, with the average workload metric, it causes 10% violation rate, while generates zero violation with peak workload metric.

6.2.7 *Clustering Results for MIP+Cluster*

We show the clustering results of Algorithm 1 in Table 6. Specifically, we set the expected cluster size $\beta = 200$, which is a moderate input size for MIP, such that it can compute the solution within a reasonable time. We evaluate its clustering quality from three perspectives, namely the independence between clusters (aggregation quality), the balance of the partition, and clustering efficiency, as we have explained in Section 5.1.

Fig. 7 visualizes the result of Algorithm 1 with all the 3,042 base stations. Overall, the result satisfies our aggregation quality requirement, as there are clear borders between the clusters. In Table 6, we list the number of BSs in each cluster with the total BS number from 1,500 to 3,042. The smallest and **largest** clusters are underlined and **bold** respectively. With the threshold β = 200, the cluster size can be controlled within the range of 150 to 400 BSs, which guarantees that the problem can be solved by MIP within an acceptable time range. As shown in Fig. 2c, even with the input of all the BSs, the problem can be solved in an acceptable time (about 90 minutes). Moreover, we list the clustering time (i.e., Time) in the last column of Table 6. It can be solved in 6.21 seconds with 1500 BSs, and less than 1 min with all the 3,042 BSs, which can be deemed as efficient in comparison to the running time of MIP.

6.2.8 *Ablation Study on τ for DA-CFS*

Table 7 demonstrates how τ (i.e., the maximum number of BS added into the candidate set from each EN's selection) affects the performance of DA-CFS. We conduct experiments with different τ values ranging from 6 to 26. It can be seen that the number of selected ENs is relatively low and stable when we set $\tau \geq 16$. In terms of its effect on the deployment cost, 22 is an elbow point as it shows a sharp decrease in the deployment cost (from 346.8k to 235.9k).

To alleviate the computation resource requirement, we always select the BS that are more distant from their originally assigned ENs as candidates. If τ is small, although the selected BS can release more computation capacity pressure from their previously assigned ENs, these BS might not be advantageous in covering more unassigned BS. In addition, a small τ value leaves less flexibility for the BS assignment. In contrast, an unreasonably high τ value would cause that the candidates might not contribute to either BS coverage or computation capacity relaxation of ENs. Therefore, we set τ to 22 in DA-CFS.

6.3 Summary

We summarise our experimental findings:

- **Cluster-based MIP.** While MIP shows a great advantage in the cost-saving objective, it is not scalable due to its efficiency. Thus, for small datasets, MIP is always the best choice, while it would not be recommended when the solution space is large. The cluster-based MIP sacrifices some accuracy to improve efficiency. Specifically, due to the high effectiveness of MIP, cluster-based MIP still greatly outperforms CFS and DA-CFS in effectiveness. With

the partition strategy, the running time has been decreased into an acceptable range compared with MIP. However, if the size of dataset further increases, more clusters will be introduced and inevitably cause longer running time as well as further decrease its effectiveness.

- **DA-CFS, CFS-F, and DA-CFS-F.** CFS baseline and DA-CFS have strengths on the efficiency, but weakness on the effectiveness. Although DA-CFS optimizes CFS on the EN selection strategy, the generated cost is still higher than MIP-based methods. However, with the fine-grained optimization to delicately address the resource allocation in the user request level, the cost generated can be decreased into a competitively low level. DA-CFS-F is undoubtedly the champion in our experiment, achieving remarkable advantages in all the perspectives, i.e., the effectiveness, the efficiency and the scalability.

7 Conclusion

In this paper, we define a Cost Minimization in MEC Edge Node Placement Problem to address the trade-off between deployment cost and users' delay tolerance. Within this problem, we define a practical and delicate delay measurement and propose a peak workload metric. We prove this problem to be NP-hard and propose a range of approximate solutions, including Cluster-based MIP, DA-CFS, and fine-grained optimization for the CFS-based methods, that can efficiently and effectively handle large-scale real-world use cases. Extensive experiments are conducted on a large real-world dataset. Our proposed solutions show significant advantages on scalability compared with the existing solutions including MIP and CFS.

In the future, we hope to further explore the methods to improve the estimation of the workload for the fine-grained case by learning the mobility pattern within those ample user requests data. It provides the opportunity to accurately estimate and foresee the workload of base stations, and thus can more delicately allocate the resources. Furthermore, we will address inter-server communication within the edge node more precisely, resulting in a more accurate workload estimation.

Acknowledgments

This research was supported partially by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (projects DP220101434 and DP220101823).

References

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

[2] 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, 2017.

[3] L. Ma, J. Wu, L. Chen, and Z. Liu, "Fast algorithms for capacitated cloudlet placements," in *CSCWD*. IEEE, 2017, pp. 439–444.

[4] J. Meng, W. Shi, H. Tan, and X. Li, "Cloudlet placement and minimum-delay routing in cloudlet computing," in *BIGCOM*. IEEE, 2017, pp. 297–304.

[5] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parall. Distr.*, vol. 27, no. 10, pp. 2866–2880, 2015.

[6] X. Zichuan, L. Weifa, X. Wenzheng, J. Mike, and G. Song, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parall. Distr.*, vol. 27, no. 10, pp. 2866–2880, 2015.

[7] L. Chen, J. Wu, G. Zhou, and L. Ma, "Quick: Qos-guaranteed efficient cloudlet placement in wireless metropolitan area networks," *J. Supercomput.*, vol. 74, no. 8, pp. 4037–4059, 2018.

[8] M. Sourav, D. Goutam, and W. Elaine, "Ccompassion: A hybrid cloudlet placement framework over passive optical access networks," in *INFOCOM*. IEEE, 2018, pp. 216–224.

[9] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurr. Comput. Pract. Exp.*, vol. 29, no. 16, p. e3975, 2017.

[10] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, no. 1, p. 32, 2019.

[11] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Softw. Pract. Exp.*, vol. 50, no. 5, pp. 489–502, 2020.

[12] S. K. Kasi, M. K. Kasi, K. Ali, M. Raza, H. Afzal, A. Lasebae, B. Naeem, S. u. Islam, and J. J. P. C. Rodrigues, "Heuristic edge server placement in industrial internet of things and cellular networks," *IEEE Internet Things*, vol. 8, no. 13, pp. 10 308–10 317, 2021.

[13] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *EDGE*. IEEE, 2018, pp. 66–73.

[14] B. Li, P. Hou, H. Wu, R. Qian, and H. Ding, "Placement of edge server based on task overhead in mobile edge computing environment," *Trans. Emerg. Telecommun. Technol.*, p. e4196, 2020.

[15] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C. Hsu, "Edge server placement in mobile edge computing," *J. Parall. Distr. Comput.*, vol. 127, pp. 160–168, 2019.

[16] Y. Hao, Z. Xu, L. H. H, L. Yan, T. Chen, Z. Shuoyao, and L. Feng, "Edge provisioning with flexible server placement," *IEEE Trans. Parall. Distr.*, vol. 28, no. 4, pp. 1031–1045, 2016.

[17] X. Xu, Y. Xue, L. Qi, X. Zhang, S. Wan, W. Dou, and V. Chang, "Load-aware edge server placement for mobile edge computing in 5g networks," in *ICSOC*. Springer, 2019, pp. 494–507.

[18] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Mobile edge computing network control: Tradeoff between delay and cost," in *GLOBECOM*. IEEE, 2020, pp. 1–6.

[19] F. Qiang and A. Nirwan, "Cost aware cloudlet placement for big data processing at the edge," in *ICC*. IEEE, 2017, pp. 1–6.

[20] "5g base station deployments," https://techblog.comsoc.org/2020/08/07/5g-base-station-deployments-open-ran-competition-huge-5g-bs-power-problem/, accessed: 2022-01-20.

[21] X. Zhang, S. Huang, H. Dong, and Z. Bao, "Edge node placement with minimum costs: When user tolerance on service delay matters," in *ICSOC*. Springer, 2021, pp. 765–772.

[22] D. Bhatta and L. Mashayekhy, "Generalized cost-aware cloudlet placement for vehicular edge computing systems," in *CloudCom*, 2019, pp. 159–166.

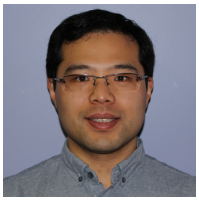
[23] Z. He, K. Li, and K. Li, "Cost-efficient server configuration and placement for mobile edge computing," *IEEE Trans.*

Parall. Distr., vol. 33, no. 9, pp. 2198–2212, 2021.
[24] X. Jiang, P. Hou, H. Zhu, B. Li, Z. Wang, and H. Ding, "Dynamic and intelligent edge server placement based on deep reinforcement learning in mobile edge computing," *Ad Hoc Netw.*, vol. 145, p. 103172, 2023.
[25] S. Liu, Y. Yu, X. Lian, Y. Feng, C. She, P. L. Yeoh, L. Guo, B. Vucetic, and Y. Li, "Dependent task scheduling and offloading for minimizing deadline violation ratio in mobile edge computing networks," *IEEE J. Sel. Area. Commun.*, vol. 41, no. 2, pp. 538–554, 2023.
[26] W. Zhou, L. Fan, F. Zhou, F. Li, X. Lei, W. Xu, and A. Nallanathan, "Priority-aware resource scheduling for uav-mounted mobile edge computing networks," *IEEE Trans. on Veh. Technol.*, vol. 72, no. 7, pp. 9682–9687, 2023.
[27] R. Yadav, W. Zhang, I. A. Elgendy, G. Dong, M. Shafiq, A. A. Laghari, and S. Prakash, "Smart healthcare: Rl-based task offloading scheme for edge-enable sensor networks," *IEEE Sen. J.*, vol. 21, no. 22, pp. 24 910–24 918, 2021.
[28] H. Taub and D. L. Schilling, *Principles of communication systems*. McGraw-Hill Higher Education, 1986.
[29] G. Li, J. Wang, J. Wu, and J. Song, "Data processing delay optimization in mobile edge computing," *Wirel. Commun. Mobile Comput.*, vol. 2018, 2018.
[30] L. Kish and C. Granqvist, "Noise in nanotechnology," *Microelectron. Reliab.*, vol. 40, no. 11, pp. 1833–1837, 2000.
[31] A. S. Tanenbaum, D. Wetherall *et al.*, "Computer networks," pp. I–XVII, 1996.
[32] J. Alber, M. R. Fellows, and R. Niedermeier, "Polynomial-time data reduction for dominating set," *J. ACM*, vol. 51, no. 3, pp. 363–384, 2004.
[33] D. Wagner and F. Wagner, "Between min cut and graph bisection," in *MFCS*. Springer, 1993, pp. 744–750.
[34] D. Eppstein, "Fast hierarchical clustering and other applications of dynamic closest pairs," *J. Exp. Algorithmics*, vol. 5, pp. 1–es, 2000.
[35] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, 2021.
[36] R. Zhou, S. Shao, W. Li, and L. Zhou, "How to define the user's tolerance of response time in using mobile applications," in *IEEM*. IEEE, 2016, pp. 281–285.

Xiaoyu Zhang received the Master degree in Information Technology from University of Melbourne. She is currently a Ph.D student in RMIT University. Her current research interests include data mining and combinatorial optimization.

Shixun Huang received the Bachelor degree from Nanjing University, Master Degree from University of Melbourne and Ph.D. degree from RMIT University. He is currently a Lecturer at University of Wollongong. His current research interests span across combinatorial optimization and mining in graph data.

Hai Dong is a senior lecturer at School of Computing Technologies in RMIT University, Melbourne, Australia. He received a Ph.D from Curtin University of Technology, Australia and a Bachelor's degree from Northeastern University, China. His research interests include Service Oriented Computing, Distributed Systems, Cyber Security, and Machine Learning. He is a senior member of the IEEE.



Zhifeng Bao received the Ph.D. degree in computer science from the National University of Singapore. He is currently a Professor with the RMIT University and an Honorary Senior Fellow with The University of Melbourne. His current research interests span across big data management and algorithm.

Jiajun Liu is a Principal Research Scientist at CSIRO's Data61, Australia. He received his Ph.D/BEng from the University of Queensland, Australia, and Nanjing University, China, in 2013 and 2006, respectively. His research focuses on Machine Learning, in particular in Multimedia Analytics and Graph Learning. Recently he is also driving a strong research effort for Edge AI-related topics, such as Knowledge Distillation, Neural Architecture Search, and Distributed Learning.

Xun Yi is currently a Professor in Cyber Security with School of Computing Technologies, RMIT University, Australia. His research interests include Cloud and IoT Security and Privacy, Distributed System Security, Blockchain Applications, and Applied Cryptography. Currently, he is an Associate Editor with IEEE Transactions on Knowledge and Data Engineering. He has been an ARC College Expert from 2017 to 2019.