Service-Aware Cooperative Task Offloading and Scheduling in Multi-access Edge Computing Empowered IoT

Zhiyan Chen[1(✉)], Ming Tao[1], Xueqiang Li[1], and Ligang He[2]

[1] School of Computer Science and Technology, Dongguan University of Technology, Dongguan, China
z.chen.8@warwick.ac.uk, {taom,lixq}@dgut.edu.cn
[2] Department of Computer Science, University of Warwick, Coventry, UK
ligang.he@warwick.ac.uk

Abstract. Multi-access edge computing(MEC) enables computation task offloading and data processing at close proximity to provide rich end-users services with ultra-low latency in Internet of things(IoT). However, the high heterogeneity of the edge node configuration and the diversity of services pose challenges in fully utilizing the computing capacity in MEC. In this paper, we consider the problem of service-aware cooperative task offloading and scheduling in a three-tier MEC empowered IoT where the service requests from IoT devices can be distributed among edge nodes or further offloaded to remote cloud. As this problem is proven to be NP-hard, we proposed a two-layer Cooperative workload Initialization and Distribution Algorithm (CIDA) to solve the problem with low time complexity by decomposing it into two subproblems: 1) the optimization problem of offloading profile under dynamic resource allocation determined by the workload type, and 2) optimization problem of computation resources allocation under given offloading profile. Extensive experiments demonstrate that CIDA achieves superior performance compared to other approaches and scales well as the system size increases.

Keywords: Multi-access edge computing · Services-aware offloading · Task scheduling · Resource allocation

1 Introduction

The Internet of Things (IoT) has been introduced to connect and coordinate the rapidly growing number of smart devices over the past decade [1,2]. And cloud computing enables IoT devices to enhance their computational capabilities and extend battery life by offloading the computation requests to cloud servers for execution. However, the extensive connectivity and data exchange in the IoT have led to substantial communication latency, which poses challenges in guaranteeing the quality of IoT services [3]. As a result, edge computing has emerged as a promising paradigm to address these challenges. By deploying computing resources at the edge where data is generated, computation requests

Z. Tari et al. (Eds.): ICA3PP 2023, LNCS 14488, pp. 327–346, 2024.
https://doi.org/10.1007/978-981-97-0801-7_19

can be processed at the closer edge node instead of offloading to the remote cloud, which significantly reduce the latency and save energy [4].

Compared to cloud servers with abundant computational resources, edge server is generally resource-limited. Severe resource competition and data processing congestion could be caused by the offloading of computation intensive requests without cooperation. Furthermore, the absence of mutual collaboration among edge servers can result in certain servers being underutilized while popular edge servers are experiencing an overwhelming computational workload [5]. Additionally, given the diverse range of services in the IoT, the requirements on response time and computation resources can vary significantly. Both processing computation-intensive requests on relatively low-capacity edge servers and the improper allocation of resources on edge servers can lead to a degradation in system performance. Therefore, the establishment of collaborative offloading and scheduling mechanisms between edge servers and remote cloud servers is crucial to fully utilize the computational resources and achieve efficient operation of the system.

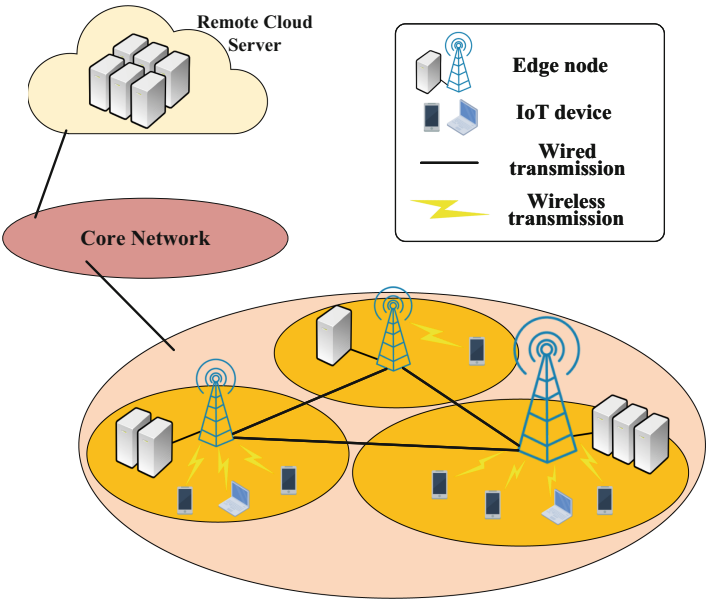

Fig. 1. Three-tier multi-access edge computing architecture

In this paper, we consider a multi-access edge computing empowered IoT system with N IoT devices, M base stations and a remote cloud server, which forms a three-tier architecture as shown in Fig. 1. Each base station is seamlessly integrated with an edge server, which is referred to as an edge node. IoT devices are connected to the edge nodes via wireless channels, while neighbouring edge nodes

establish connectivity through local area networks or wired peer-to-peer links [6]. Furthermore, edge nodes possess the capability to further offload computation tasks to remote cloud server via the core network. In such system, individual IoT device may requests multiple types of services. And each specific service request made by an IoT device can be handled in one of the following ways: offloaded to its associated edge node for processing, transferred to another edge node within the network for processing, or further offloaded via the core network to remote cloud server with much higher computing capacity. Different types of services on an IoT device may offloaded to different destination for processing to improve the system performance, which can be achieved by a centralized offload controller such as SDN controller [7,8]. However, ensuring efficient system operation and resource utilization in such system presents a significant challenge as it requires an appropriate offloading and computing capacity allocation profiles to be obtained at low time cost. The close interplay of these profiles and the large searching space further complicate the problem.

To solve this problem, we discuss the design and implementation of CIDA, which is a two-layer heuristic Cooperative workload Initialization and Distribution Algorithm. The first layer of CIDA initialize the system offloading profile with the transmission delays for system requests are minimized regardless of the capacity constraint of edge nodes, then CIDA iteratively update the offloading profile by reallocating the requests to the execution platform with their response time are minimized. The second layer of CIDA obtains the optimal computing capacity allocation at each edge nodes by solving the KKT conditions based on the offloading profile in the first layer.

To summarize, we have made the following contributions in this paper:

(1) The problem of cooperative service-aware task offloading and scheduling in a three-tier multi-access edge computing system is investigated, which is formulated as an joint optimization problem of offloading profile and computing capacity allocation profile.
(2) The NP-hardness of the joint optimization problem is proven by demonstrating a reduction from the well-known NP-hard makespan scheduling problem to a specific case of the joint optimization problem.
(3) We design a two-layer heuristic algorithm CIDA to solve the joint optimization problem at a low time complexity by decompose it into two subproblems. In the first layer, an offloading profile is derived base on allocating the service requests to the execution platform with minimum response time iteratively. In the second layer, as the convexity of computing capacity allocation problem in each edge node is proved, the optimal computing capacity allocation profile could be obtained.

The rest of the paper is organized as follows. We review the related works in Sect. 2. Section 3 presents the overall system and the joint optimization problem formulation. In Sect. 5, the design and implementation of algorithm CIDA is presented. The results of the numerical simulations are provided in Sect. 6. Finally, this paper is concluded in Sect. 7.

2 Related Work

To fully utilize the potential of multi-access edge computing, the problem of tasks offloading under various scenarios attracted much attention of the research community lately. One major objective is to reduce the task completion latency and energy consumption [9,10]. In [9], an optimization algorithm based on genetic algorithm and particle swarm optimization is proposed to minimize the energy consumption of the system in a densely deployed distributed small cell network. In [10], a distributed association scheme is proposed to minimize the latency of data flows in a fog computing IoT system by iteratively associating IoT devices to suitable base station until convergence. Zhu et al. [11] addressed the triple-objective optimization problem of energy cost, task completion delay and network deployment cost. A metaheuristic algorithm based on whale optimization is adopted to produce a set of Pareto-optimal solutions for the problem. Besides the offloading profile optimization, the computation resource allocation optimization has been investigated in some research [12]. A cooperative offloading model based on software defined network is proposed in [12] for optimizing task scheduling in LTE-advanced networks. And [13] proposed an UAV-aided framework in federated-WSN-enabled IoT for trust-worthy data collection. However, these literature mainly emphasize the offloading between devices and the edge servers, without considering the further offloading from edge servers to the remote cloud.

There are also a few work done on the cooperation workload scheduling in the heterogeneity three-tier MEC network [14–17]. [16] studies the online workload scheduling among edge-clouds architecture and introduced an online job dispatching and scheduling algorithm to minimized the job response time. [18] proposed a joint optimization approach to allocate the computation resource to IoT devices in a three-tier fog IoT framework. In [17], the offloading decisions, communication resources and computation resources are taken into account to minimize the overhead in a three-tier MEC system. [19] focus on minimizing the energy cost and allocating the computation resource with the latency deadlines of mobile devices are satisfied. Three algorithms are presented to solve the problem under different time complexity performance. Nevertheless, all the above work have overlooked that an IoT device might request multiple type of services. Given the heterogeneous nature of service applications deployed on IoT devices, incorporating the consideration of various types of service requests within the same device becomes imperative when optimizing task scheduling in the IoT. Such an approach facilitates the provision of enhanced and finer-grained offloading strategies for MEC system.

3 System Model and Problem Formulation

For the sake of readability, we summarize the symbols used in Table 1. Denote $\mathcal{N} = \{1, 2, 3, ..., N\}$ as the set of IoT devices and $\mathcal{M} = \{1, 2, 3, ..., M\}$ as the set of edge nodes. Denote $\mathcal{J} = \{1, 2, 3, ..., J\}$ as the set of services provided in the system, and $\mathcal{J}_i$ as the set of services on IoT device i. The service request of type j is described as (L_j, C_j), where L_j represents the average input data

length associated with the request, and C_j represents the average computation requirements of the corresponding task (in CPU cycles) for type j services. As in other paper [20], we assume the arrival of each type of service requests on IoT devices follows a Poisson process with λ_{ij} denotes the expect arrival rate of type j service requests on IoT device i.

Table 1. Summary of symbols

Symbol	Definition
$\mathcal{N}$	The set of IoT devices
$\mathcal{M}$	The set of edge nodes
$\mathcal{J}$	The set of services provided in the system
$\mathcal{J}_i$	The set of services on IoT device i
L_j	Average input data length associate with type j service request
C_j	Average computation requirements for type j service request
λ_{ij}	Expected arrival rate on type j service request on IoT device i
a^e_{ijk}	Indicator to denote if j request on device i is executed on edge node k
a^c_{ij}	Indicator to denote if j request on device i is executed on remote cloud
b_i	Associated edge node of IoT device i
T^e_{ijk}	Average response time for j request on device i executed on edge node k
T^{trans}_{i,b_i}	Transmission delay from IoT device i to its associated edge node b_i
$T^{trans}_{b_i,k}$	Transmission delay from edge node b_i to edge node k
T^{exe}_{ijk}	Execution delay for j request on device i to be executed on edge node k
R_{i,b_i}	Data rate achieved for device i to associated edge node b_i
B_{i,b_i}	Wireless channel bandwidth between device i to associated edge node b_i
P_i	Transmission power of IoT device i
G_{i,b_i}	Channel gain between device i to associated edge node b_i
σ^2_i	Noise power of device i
$\bar{L}_i$	Average input data length of the requests on device i
f_k	Computing capacity of edge node k
f_{jk}	Computing capacity allocated to type j request on edge node k
T^c_{ij}	Average response time for j request on device i executed on remote cloud
$T^{trans}_{b_i,c}$	Transmission delay from edge node b_i to remote cloud
$A(ij)$	Indicator of execution platform for type j service requests on IoT device i
$\boldsymbol{A}$	Offloading profile for all the requests on IoT devices in the system
$\boldsymbol{F}_k$	Computing capacity allocation profile on edge node k
$\boldsymbol{F}$	Computing capacity allocation profile for all edge nodes in the system
$T_{ij}(\boldsymbol{A}, \boldsymbol{F})$	Average response time for type j request on device i under profile $\boldsymbol{A}$ and $\boldsymbol{F}$
ΔT_{ij}	Improvement on average response time for type j request on device i by moving it to its optimal edge node
$\mathcal{R}$	The set of remaining requests to be reallocated in the system
r_{ij}	The index of the type j request on device i in the set $\mathcal{R}$

Let a^e_{ijk} and a^c_{ij} be the binary indicator ($a^e_{ijk}, a^c_{ij} \in \{0,1\}$) which denote if type j service requests of IoT device i to be executed on edge node k or the remote cloud server, respectively. A service request is either be processed on an edge node or the remote cloud server, thus we have

$$\sum_{k \in \mathcal{M}} a^e_{ijk} + a^c_{ij} = 1, i \in \mathcal{N}, j \in \mathcal{J}_i \tag{1}$$

3.1 Edge Computing Model

For each IoT device in the system, it is associated to its closest edge node to offloading the service requests. Note that the associated edge node might not be the eventual execution platform for the requests on IoT devices. Denote b_i as the associated edge node of IoT device i. When type j requests on IoT device i are to be executed on edge node k, the total response time consist of the following four parts: the time it takes for the requests to be transferred from IoT device i to its associated edge node b_i (denoted by T^{trans}_{i,b_i}), the time it takes for the requests to be transferred from associated edge node b_i to the execution edge node k (denoted by $T^{trans}_{b_i,k}$), the time it takes for the requests to be executed on edge node k (denoted by T^{exe}_{ijk}), and the time it takes for sending the computation outcome back to IoT device i.

Similar to many studies such as [21–23], we omit the time overhead associated with the transmission of computation outcomes back to IoT device, which is based on the observation that the length of computation outcomes tends to be significantly smaller compared to the length of input data for numerous services such as image recognition and speech recognition. Hence, the total response time of type j requests on IoT device i to be executed on edge node k is given by

$$T^e_{ijk} = T^{trans}_{i,b_i} + T^{trans}_{b_i,k} + T^{exe}_{ijk} \tag{2}$$

During the wireless transmission between IoT device i and its associated edge node b_i, the data rate R_{i,b_i} can be generally expressed as a logarithmic function according to the Shannon theorem. Denote B_{i,b_i} as the channel bandwidth between IoT device i and edge node b_i, G_{i,b_i} as the channel gain between them, P_i as the transmission power and σ_i^2 as the noise power of IoT device i. The data rate of IoT device i can be expressed as

$$R_{i,b_i} = B_{i,b_i} \log(1 + \frac{P_i G_{i,b_i}}{\sigma_i^2}). \tag{3}$$

As mentioned earlier that the arrival of each type of service request on IoT devices is a Poisson process, the arrival of all the requests on IoT device i, which is a sum of multiple independent Poisson processes, follows a Poisson process with expected rate $\sum_{j \in \mathcal{J}_i} \lambda_{ij}$. Thus, the offloading from IoT device i to its associated edge node b_i can be modelled as an $M/M/1$ queue, and the average response time is given by

$$T^{trans}_{i,b_i} = \frac{1}{R_{i,b_i}/\bar{L}_i - \sum_{j \in \mathcal{J}_i} \lambda_{ij}} \tag{4}$$

where $\bar{L}_i$ denotes the average input data length for the service requests on IoT device i and $\bar{L}_i = \sum_{j\in\mathcal{J}_i} L_j\lambda_{ij} / \sum_{j\in\mathcal{J}_i} \lambda_{ij}$. To keep the queue stable we have $R_{i,b_i}/\bar{L}_i > \sum_{j\in\mathcal{J}_i} \lambda_{ij}$.

After the requests being offloaded to associated edge node b_i, they will be transferred to the execution edge node k. The transmission delay from b_i to k is denoted by $T^{trans}_{b_i,k}$, which can be measured and recorded by the SDN controller [24]. Note that $T^{trans}_{b_i,k} = 0$ when $b_i = k$. According to [25], the transmission delay between edge nodes is modelled as a linear function of the distance between them, which is given by

$$T^{trans}_{\mathrm{b_i},k} = \alpha \cdot d_{b_i,k} + \beta \tag{5}$$

where $d_{b_i,k}$ denotes the distance between edge node b_i and edge node k. And typically, $\alpha = 5$ and $\beta = 22.3$.

When type j requests on IoT device i are transferred to edge node k for execution, it shares the computing capacity allocated to type j requests on edge node k with same type of requests which are offloaded from other IoT devices. Given that the offloading of individual IoT device follows a Poisson process, it can be inferred that the arrival of each type of service request on the edge node also exhibits a Poisson process. The processing for each type of requests on the edge nodes can be modelled as an $M/M/1$ queue. Denote f_k as the computing capacity of edge node k and f_{jk} as the computing capacity allocated to type j request on edge node k. The computation delay for type j requests of IoT device i on edge node k is calculated as

$$T^{exe}_{ijk} = \frac{1}{f_{jk}/C_j - \sum\limits_{i\in\mathcal{N}} a^e_{ijk}\lambda_{ij}}. \tag{6}$$

To ensure the queue is stable, we have $\sum_{j\in\mathcal{J}} f_{jk} \le f_k$ and $f_{jk}/C_j > \sum_{i\in\mathcal{N}} a^e_{ijk}\lambda_{ij}$.

3.2 Cloud Computing Model

For the service requests on IoT device i which are offloaded to remote cloud server for execution, the total response time consist of following four parts: the time it takes for the requests to be transferred from IoT device i to its associated edge node b_i, the time it takes for the requests to be transferred from edge node b_i to the remote cloud server via the core network, the time it takes for processing the requests on the remote cloud, and the time it takes for sending the computation results back to device i.

Similar to the edge computing model in Sect. 3.1, we disregarded the transmission delay of computation results from remote cloud to the IoT devices. Additionally, as the computing capacity of cloud servers exceeds that of edge nodes significantly, and the primary source of delay in cloud computing is attributed to data offloading from IoT devices to the remote cloud server, we have omitted the service execution delay on the cloud. Thus, denote $T^{trans}_{b_i,c}$ as the transmission delay between edge node b_i and the remote cloud, the total response time of

type j requests on IoT device i to be executed on remote cloud server is given by

$$T_{ij}^{c} = T_{i,b_i}^{trans} + T_{b_i,c}^{trans}. \tag{7}$$

4 Problem Formulation

In this section, we present the problem formulation for service-aware task offloading and scheduling, focusing on the minimization of the average response time for all service requests. Denote $A(ij)$ as the indicator of execution platform for type j service requests on IoT device i, which is defined as $A(ij) = \{a_{ijk}^{e}, a_{ij}^{c} | k \in \mathcal{M}\}$. And $\boldsymbol{A}$ denotes the offloading profile for all the service requests on IoT devices in the system, which is defined as $\boldsymbol{A} = \{A(ij) | i \in \mathcal{N}, j \in \mathcal{J}_i\}$. Furthermore, Denote $\boldsymbol{F}_k$ as the computing capacity allocated to each type of service on edge node k and $\boldsymbol{F}_k = \{f_{jk} | j \in \mathcal{J}\}$. And $\boldsymbol{F}$ denotes the computing capacity allocation for all the edge nodes, which is given by $\boldsymbol{F} = \{\boldsymbol{F}_k | k \in \mathcal{M}\}$. Therefore, the service-aware task offloading and scheduling problem can be formulated as the problem of minimizing the average response time for all service requests with respect to $\boldsymbol{A}$ and $\boldsymbol{F}$, which defined as:

$$\mathbf{P1}: \min_{\boldsymbol{A},\boldsymbol{F}} \quad \sum_{i\in\mathcal{N}, j\in\mathcal{J}_i} \frac{\lambda_{ij}}{\sum_{i\in\mathcal{N}, j\in\mathcal{J}_i} \lambda_{ij}} \left(\sum_{k\in\mathcal{M}} a_{ijk}^{e} T_{ijk}^{e} + a_{ij}^{c} T_{ij}^{c} \right) \tag{8}$$

$$s.t \quad \sum_{k\in\mathcal{M}} a_{ijk}^{e} + a_{ij}^{c} = 1, i \in \mathcal{N}, j \in \mathcal{J}_i \tag{9}$$

$$a_{ijk}^{e}, a_{ij}^{c} \in \{0, 1\}, i \in \mathcal{N}, j \in \mathcal{J}_i, k \in \mathcal{M} \tag{10}$$

$$R_{i,b_i} / \bar{L}_i > \sum_{j\in\mathcal{J}_i} \lambda_{ij}, i \in \mathcal{N} \tag{11}$$

$$f_{jk} / C_j > \sum_{i\in\mathcal{N}} a_{ijk}^{e} \lambda_{ij}, j \in \mathcal{J}, k \in \mathcal{M} \tag{12}$$

$$\sum_{j\in\mathcal{J}} f_{jk} \leq f_k, k \in \mathcal{M} \tag{13}$$

$$f_{jk} \in [0, f_k], j \in \mathcal{J}, k \in \mathcal{M} \tag{14}$$

The constraints (9) and (10) delineate the processing of each type of service requests on IoT devices are either on an edge node or on the remote cloud server. The constraint (11) ensures that the total traffic data on each IoT device remains within the wireless channel capacity between the device and its associated edge node. The constraint (12) guarantees the workload assigned to each type of requests on each edge node does not exceed the specific computing capacity allocated to that request type. The constraint (13) ensures that the collective computing capacity assigned to each type of request within an edge node should not exceed the overall computing capacity of the edge node. Lastly, constraint (14) enforces that the computing capacity allocated to each request type remains within the total computing capacity available of the edge node.

However, solving problem **P1** optimally with low searching cost could be quite challenging. The typical size of edge computing models tends to be substantial, which results in a large searching space that optimization algorithm need to explore. Additionally, the high heterogeneity on the configuration of edge nodes and the diversity of services on IoT devices further complicate the problem. Moreover, the interplay between the IoT devices offloading profile $\boldsymbol{A}$ and the edge node capacity allocation profile $\boldsymbol{F}$ must be considered during the optimization process, making problem **P1** hard to tackle. We will provide a proof of its NP-hardness to demonstrate the computational complexity of problem **P1** next.

Theorem 1. *Problem* **P1** *is at least NP-hard.*

Proof. Here, we present an analysis of a simplified case of problem **P1** to demonstrate its NP-hardness. In this particular scenario, only a single type of service is considered in the system. And we make the following assumptions: the transmission delay from IoT devices to their associated edge node are assumed to be 0, i.e., $T^{trans}_{i,b_i} = 0$; the transmission delay between edge nodes are assumed to be 0, i.e., $T^{trans}_{b_i,k} = 0$; the transmission delay from edge nodes to the remote cloud server are assumed to be infinite, i.e., $T^{trans}_{b_i,c} = \infty$, which indicates an infinite response time for requests executed on the remote cloud according to Equation (7). As a result, all the requests should be processed on edge nodes if capable.

By making these assumptions, problem **P1** is reduced to the classic makespan scheduling problem. In this transformation, the service requests on each IoT device are projected as jobs to be scheduled, while the edge nodes are projected as the processing machines in the makespan scheduling problem. As the classic makespan scheduling problem, which requires exploring all possible combinations of job allocation to find the optimal solution, has non-polynomial computation complexity for even there are only two identical processing machines, the NP-hardness of problem **P1** is established.

5 Algorithm Design

As stated above, even the simplified case of problem **P1** remain challenging to be solved in polynomial time complexity. As we can observed that problem **P1** involve a jointly optimization of the offloading profile of IoT devices $\boldsymbol{A}$ and the computing capacity allocation profile of edge nodes $\boldsymbol{F}$. To solve problem **P1**, we carefully decompose the joint optimization involved into two sub problem:

(1) The optimization problem of offloading profile $\boldsymbol{A}$ under a certain computing capacity allocation policy.
(2) The optimization problem of computing capacity allocation profile $\boldsymbol{F}$ under a specific offloading profile $\boldsymbol{A}$.

Accordingly, we present a two-layer Cooperative workload Initialization and Distribution Algorithm (CIDA) with the first layer minimizing the transmission delay and distributing the workload among edge nodes and the cloud server to

obtain a suboptimal offloading profile, and the second layer allocate computing capacity to each type of services optimally on the edge nodes based on the offloading profile obtained in the first layer.

5.1 Offloading Profile Optimization Problem

To tackle the optimization problem of offloading profile, CIDA follows a two-step approach. Firstly, the service requests on IoT devices are assigned to their respective associated edge nodes with the transmission delay is minimized, establishing the initial offloading profile. Subsequently, an iterative process is performed to search for and reallocate requests which has the highest response time improvement by moving them from the current execution platform to the optimal execution platform. This process continues until no further improvement in response time can be achieved for any remaining requests in the system.

Offloading Profile Initialization. To initialize the offloading profile, we first focus on minimizing the average transmission delay from IoT devices to corresponding execution edge node for all the requests, which is given by

$$T_{ij}^{trans} = \sum_{k \in \mathcal{M}} a_{ijk}^{e}(T_{i,b_i}^{trans} + T_{b_i,k}^{trans}) + a_{ij}^{c}(T_{i,b_i}^{trans} + T_{b_i,c}^{trans}) \tag{15}$$

Thus, the transmission delay minimization problem is formulated as

$$\mathbf{P2}: \min_{\boldsymbol{A}} \quad \sum_{i \in \mathcal{N}, j \in \mathcal{J}_i} \frac{\lambda_{ij}}{\sum_{i \in \mathcal{N}, j \in \mathcal{J}_i} \lambda_{ij}} T_{ij}^{trans} \tag{16}$$

$$s.t \quad \sum_{k \in \mathcal{M}} a_{ijk}^{e} + a_{ij}^{c} = 1, i \in \mathcal{N}, j \in \mathcal{J}_i \tag{17}$$

$$a_{ijk}^{e}, a_{ij}^{c} \in \{0,1\}, i \in \mathcal{N}, j \in \mathcal{J}_i, k \in \mathcal{M} \tag{18}$$

$$R_{i,b_i}/\bar{L}_i > \sum_{j \in \mathcal{J}_i} \lambda_{ij}, i \in \mathcal{N} \tag{19}$$

It is easy to observe that to minimize T_{ij}^{trans}, all the requests on IoT devices will be allocated to its associated edge node for execution, i.e., $A(ij) = b_i, i \in \mathcal{N}, j \in \mathcal{J}_i$, as the transmission delay of type j requests on IoT device i from associated edge node to the execution edge node $T_{b_i,k}^{trans} = 0$ when $b_i = k$, and $T_{b_i,b_i}^{trans} < T_{b_i,k}^{trans}$ holds when $b_i \neq k$ according to Equation (5)(15). Denote $\boldsymbol{A}_{ini}$ as the corresponding offloading profile, i.e., $\boldsymbol{A}_{ini} = \{A(ij) = b_i | i \in \mathcal{N}, j \in \mathcal{J}_i\}$.

However, offloading profile $\boldsymbol{A}_{ini}$ might not be a feasible solution for problem **P1** since the following constraints might be violated:

$$\sum_{i \in \mathcal{N}, j \in \mathcal{J}_i} a_{ijk}^{e} \lambda_{ij} C_j < f_k, k \in \mathcal{M}. \tag{20}$$

Constraint (20) can be derivated from constraint (12) and (13). Specifically, allocating the requests on the IoT devices to its associated edge nodes could

minimize the transmission delay in the system. Nevertheless, due to the variation in geographical distribution of IoT devices, the edge nodes with more IoT devices are associated to will be assigned a greater computational workload which might exceed the computing capacity of the edge nodes, while the workload on other edge nodes remain at a relatively low level. To address this issue, the workload need to be reallocated among edge nodes and remote cloud server to ensure efficient utilization of computing resources and minimize total response time for all requests.

Workload Reallocation. This subsection presents the main idea of algorithm design to reallocate the requests workload in the system. We first define the computing capacity allocation profile on the edge nodes during the reallocation process. Let the computing capacity allocated to each type of request to be determined based on the percentage of corresponding type of workload in the total workload on the edge node, which is given by

$$f_{jk} = \frac{\sum\limits_{i \in \mathcal{N}} a^e_{ijk} \lambda_{ij} f_k}{\sum\limits_{i \in \mathcal{N}, j \in \mathcal{J}_i} a^e_{ijk} \lambda_{ij}}, j \in \mathcal{J}, k \in \mathcal{M} \tag{21}$$

In addition, denote $T_{ij}(\boldsymbol{A}, \boldsymbol{F})$ as the total response time of type j service of IoT device i under the current offloading profile $\boldsymbol{A}$ and computing capacity allocation profile $\boldsymbol{F}$. Note that $T_{ij}(\boldsymbol{A}, \boldsymbol{F}) = \infty$ for the requests on the edge node with constraint (20) is violated. Denote $\boldsymbol{A}_{-ij}$ as the offloading profile of the system other than type j service requests on IoT device i. If $\boldsymbol{A}_{-ij}$ is given, it can be determined that the optimal execution platform for type j service on device i to minimize its total response time. Denote $A(ij)^*$ as the indicator of optimal execution platform and $T_{ij}(A(ij)^*, \boldsymbol{A}_{-ij}, \boldsymbol{F})$ as the total response time of type j request on device i on execution platform $A(ij)^*$. Denote ΔT_{ij} as the response time improvement for reallocating type j requests on device i from current execution platform $A(ij)$ to optimal execution platform $A(ij)^*$. Thus we have

$$T_{ij}(\boldsymbol{A}, \boldsymbol{F}) = \begin{cases} \infty & a^e_{ijk} = 1, \sum\limits_{i \in \mathcal{N}, j \in \mathcal{J}_i} a^e_{ijk} \lambda_{ij} C_j < f_k \\ \sum\limits_{k \in \mathcal{M}} a^e_{ijk} T^e_{ijk} + a^c_{ij} T^c_{ij} & otherwise \end{cases} \tag{22}$$

$$A(ij)^* = \arg\min_{A(ij)} \{T_{ij}(A(ij), \boldsymbol{A}_{-ij}, \boldsymbol{F}) | i \in \mathcal{N}, j \in \mathcal{J}_i\} \tag{23}$$

$$\Delta T_{ij} = T_{ij}(\boldsymbol{A}, \boldsymbol{F}) - T_{ij}(A(ij)^*, \boldsymbol{A}_{-ij}, \boldsymbol{f}) \tag{24}$$

Through a calculation of response time improvement for the requests which has not been reallocated in the system, the requests with highest response time improvement will be put into a set and wait for update. One of them will be selected randomly and reallocated to its optimal execution platform at each iteration. If there are multiple execution platforms that can achieve the optimal

response time for a selected request, one of them is randomly chosen for reallocation. This process repeats until all the requests in the system is reallocated or no further improvement on response time can be achieve for the remaining requests in the system.

Denote r_{ij} as the index of type j request on IoT device i. Denote $\mathcal{R}$ as the set of remaining service requests to be reallocated in the system, i.e., $\mathcal{R} = \{r_{ij} | i \in \mathcal{N}, j \in \mathcal{J}_i\}$. Denote R_{update} as the set for the requests having maximum response time improvement and waiting to be updated. The first layer of CIDA is shown in Algorithm(1).

Algorithm 1. The first layer: offloading profile optimization

Input: $\{L_j, C_j | j \in \mathcal{J}\}, \{\lambda_{ij} | i \in \mathcal{N}, j \in \mathcal{J}_i\}, \{B_{i,b_i}, P_i, G_{i,b_i}, \sigma_i^2, T_{b_i,c}^{trans} | i \in \mathcal{N}\}, \alpha, \beta, \{f_k | k \in \mathcal{M}\}$

Output: A solution of offloading profile $\boldsymbol{A} = \{A(ij) | i \in \mathcal{N}, j \in \mathcal{J}_i\}$, where $A(ij) = \{a_{ijk}^e, a_{ij}^c | k \in \mathcal{M}\}$.

$A(ij) \leftarrow b_i, i \in \mathcal{N}, j \in \mathcal{J}_i$
for Iteration $j = 1, 2, 3, ...$ **do**
  $R_{update} \leftarrow \varnothing$
  $\Delta T_{max} = 0$
  for Each request $r_{ij} \in \mathcal{R}$ **do**
    Calculate $A(ij)^*$ and ΔT_{ij} by Equation (1)∼(7) and (21)∼(24)
    if $\Delta T_{ij} > \Delta T_{max}$ **then**
      $R_{update} \leftarrow \varnothing \cup \{r_{ij}\}$
      $\Delta T_{max} \leftarrow \Delta T_{ij}$
    else if $\Delta T_{ij} = \Delta T_{max} \&\& \Delta T_{max} \neq 0$ **then**
      $R_{update} = R_{update} \cup \{r_{ij}\}$
    end if
  end for
  if $R_{update} \neq \varnothing$ **then**
    Select r_{ij} in R_{update} randomly for update
    $A(ij) \leftarrow A(ij)^*$
    $\mathcal{R} = \mathcal{R} - \{r_{ij}\}$
  else
    return the offloading profile $\boldsymbol{A}$
  end if
end for

We will now prove the finite termination property of Algorithm 1. Initially, the size of the set of remaining service requests to be reallocated is $|\mathcal{R}| = \sum_{i \in \mathcal{N}} |\mathcal{J}_i|$. At each iteration, $|\mathcal{R}|$ is decreased by one, as the request with the maximum response time improvement is successfully reallocated to its optimal execution platform and removed from the set $\mathcal{R}$. Consequently, it takes $|\mathcal{R}|$ iterations to reallocate all the requests in the system, resulting in Algorithm 1 terminating when $\mathcal{R} = \varnothing$. Alternatively, Algorithm 1 may terminate earlier if there are no requests in the system that can further improve their response times through reallocation ($\Delta T_{max} = 0$).

The time complexity of Algorithm 1 is analyzed as follows: Step 1 has a time complexity of $O(|\mathcal{R}|)$, as it performs a certain operation for each request in $\mathcal{R}$. In the worst case scenario, Step 2 is iterated $|\mathcal{R}|$ times. As step 5 and step 6 involve calculating the response time of requests on all execution platforms, which have the size of $|\mathcal{R}|$ and $|\mathcal{M}+1|$, the time complexity of these steps is $O(|\mathcal{R}||\mathcal{M}+1|)$. And step 7 to step 20 have a constant time complexity of $O(1)$. Taking all these steps into account, the overall time complexity of Algorithm 1 can be approximated as $O(|\mathcal{R}|^2(\mathcal{M}+1))$.

5.2 Computing Capacity Allocation Profile Optimization Problem

With the offloading profile $\boldsymbol{A}$ is determined, the transmission delay for all the requests can be calculated. And **P1** is transformed to a computing capacity allocation problem, with the objective of minimizing the execution delay for the requests in the system. Since the execution delay for requests on one edge node is independent of requests on other edge nodes, the computing capacity allocation problem for the system can be decomposed into multiple computing capacity allocation problems for each individual edge node. Thus the allocation problem on edge node k is given by:

$$\mathbf{P3}: \min_{\boldsymbol{F}_k} \quad \sum_{i\in\mathcal{N},j\in\mathcal{J}_i} \frac{a^e_{ijk}\lambda_{ij}}{\sum\limits_{i\in\mathcal{N},j\in\mathcal{J}_i}\lambda_{ij}} T^{exe}_{ijk} \tag{25}$$

$$f_{jk}/C_j > \sum_{i\in\mathcal{N}} a^e_{ijk}\lambda_{ij}, j\in\mathcal{J} \tag{26}$$

$$\sum_{j\in\mathcal{J}} f_{jk} \le f_k \tag{27}$$

$$f_{jk}\in[0,f_k], j\in\mathcal{J} \tag{28}$$

Theorem 2. *Problem* **P3** *is a convex optimization problem over the computing capacity allocation profile* $\boldsymbol{F}$.

Proof. An optimization problem is a convex optimization problem with following conditions are satisfied: the objective function is a convex function; the constraint functions are convex functions; the feasible region is a convex set. It is easy to observed that the constraints (26), (27), (28) are convex and the feasible region is a convex set. And we can prove the objective function of problem **P3** by showing the positive definiteness of its Hessian matrix.

Denote Z_k as the objective function of problem **P3**, thus we have $Z_k = \sum\limits_{i\in\mathcal{N},j\in\mathcal{J}_i} \frac{a^e_{ijk}\lambda_{ij}}{\sum\limits_{i\in\mathcal{N},j\in\mathcal{J}_i}\lambda_{ij}} \left(\frac{1}{f_{jk}/C_j - \sum\limits_{i\in\mathcal{N}} a^e_{ijk}\lambda_{ij}}\right)$. The Hessian matrix of Z is denoted by $H = \left[h_{jj'}\right]_{j\times j'}$. When $j = j'$, $h_{jj'} = \sum\limits_{i\in\mathcal{N}} \frac{a^e_{ijk}\lambda_{ij}}{\sum\limits_{i\in\mathcal{N},j\in\mathcal{J}_i}\lambda_{ij}} \cdot \frac{2}{C_j^2(f_{jk}/C_j - \sum\limits_{i\in\mathcal{N}} a^e_{ijk}\lambda_{ij})^3}$, otherwise $h_{jj'} = 0$, meaning that H is a positive definite matrix and Z is a convex function. Therefore, Problem **P3** is a convex optimization problem over the computing capacity allocation profile $\boldsymbol{F}_k$.

As problem **P3** is a convex optimization problem, it can be solved optimally via solving Karush-Kuhn-Tucker(KKT) conditions. We first construct the Lagrange function of Problem **P3** as

$$\begin{aligned} L_k(\boldsymbol{F}_k, \boldsymbol{\chi_k}, \gamma_k, \boldsymbol{\phi_k}, \boldsymbol{\omega_k}) = Z_k - \sum_{j \in \mathcal{J}} \chi_{jk}(f_{jk} - \sum_{i \in \mathcal{N}} a^e_{ijk} \lambda_{ij} C_j) \\ + \gamma_k(\sum_{j \in \mathcal{J}} f_{jk} - f_k) + \sum_{j \in \mathcal{J}} \phi_{jk}(f_{jk} - f_k) - \sum_{j \in \mathcal{J}} \omega_{jk} f_{jk} \end{aligned} \tag{29}$$

with $\boldsymbol{\chi_k}$, γ_k, $\boldsymbol{\phi_k}$ and $\boldsymbol{\omega_k}$ are the set of Lagrange multipliers, i.e., $\boldsymbol{\chi_k} = \{\chi_{jk} | j \in \mathcal{J}\}$, $\boldsymbol{\phi_k} = \{\phi_{jk} | j \in \mathcal{J}\}$ and $\boldsymbol{\omega_k} = \{\omega_{jk} | j \in \mathcal{J}\}$.

The KKT conditions of problem **P3** is given by constraints (26) $\sim$ (28) and equations (30) $\sim$ (35).

$$\frac{\partial L_k(\boldsymbol{F}_k, \boldsymbol{\chi_k}, \gamma_k, \boldsymbol{\phi_k}, \boldsymbol{\omega_k})}{\partial f_{jk}} = 0, \quad \forall j \in \mathcal{J} \tag{30}$$

$$\chi_{jk}(f_{jk} - \sum_{i \in \mathcal{N}} a^e_{ijk} \lambda_{ij} C_j) = 0, \quad \forall j \in \mathcal{J} \tag{31}$$

$$\gamma_k(\sum_{j \in \mathcal{J}} f_{jk} - f_k) = 0 \tag{32}$$

$$\phi_{jk}(f_{jk} - f_k) = 0, \quad \forall j \in \mathcal{J} \tag{33}$$

$$\omega_{jk} f_{jk} = 0, \quad \forall j \in \mathcal{J} \tag{34}$$

$$\chi_{jk}, \gamma_k, \phi_{jk}, \omega_{jk} \geq 0, \quad \forall j \in \mathcal{J} \tag{35}$$

As the KKT conditions of problem **P3** is solved, an optimal computing capacity allocation profile of edge node k can be derived. By aggregating the allocation profiles obtained for each individual edge node $\boldsymbol{F}_k$, the optimal allocation profile for the entire system $\boldsymbol{F}$ can be acquired.

6 Simulation Results

In this section, extensive simulations are conducted to evaluate the system performance of CIDA. The simulations are implemented with a MATLAB program on the executing host: 64-bit Windows 11 operating system, Intel Core i5-9600K CPU@3.70 GHz, and 32GB of RAM. In the simulations, 1000 IoT devices and 50 edge nodes are randomly distributed in a 200 km $\times$ 200 km area. The system offers 10 types of services in total, and each IoT device receives five random service requests out of the available 10. Unless otherwise stated, the default value of parameters are specified in Table 2.

We will start by evaluating the convergence of CIDA. Figure 2 illustrates the average response time of all service requests in the system over iterations. It can be observed that the system average response time prior to the 877th iteration remains infinite. This can be attributed to the initial allocation of requests to the associated edge nodes without considering the capacity of these

Table 2. Configuration of Simulation Parameters

Parameter	Value/Range	Parameter	Value/Range
λ_{ij}	$[2, 3]$	G_{i,b_i}	$128.1 + 37.5 \log d_{i,b_i}$
L_j	$[80, 120]$KB	B_{i,b_i}	1MHz
C_j	$[10, 15]$ MegaCycles	P_i	23dBm
σ_i^2	-100dBm	$T_{b_i,c}^{trans}$	0.5 s
f_k	$[5.4, 6.6]$GHz		

nodes. Consequently, requests on overwhelmed edge nodes experience an infinite execution delay, leading to an overall average response time of infinity for the system. However, after the 877th iteration, there is a significant decrease in the system average response time, and it eventually converges at the 2480th iteration as there are no requests left that can further improve the system's response time. Therefore, this observation verifies the convergence of CIDA.

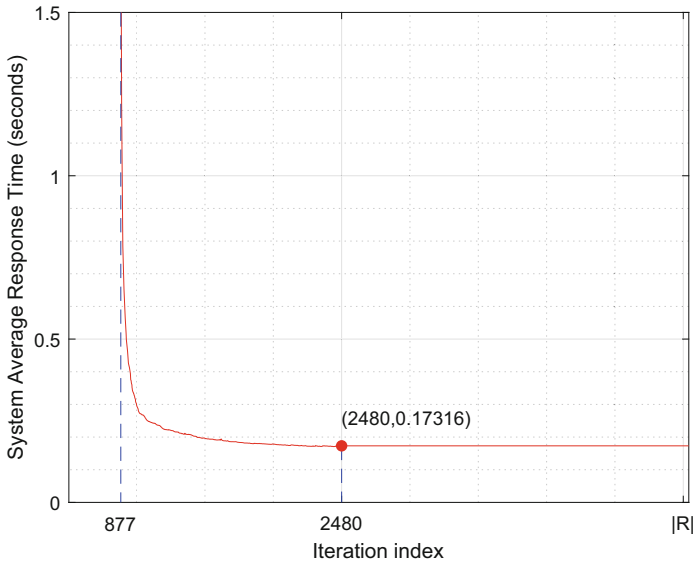

Fig. 2. The system average response time over iterations

We implemented the following algorithms as benchmark for comparison purpose:

(1) **Density Based Clustering Strategy (DBCS)**: All the requests will be offloaded to the edge node with minimum average response time until the workload on the edge node exceed the average workload among all the edge nodes, as in [26].

(2) **Distributed Offloading Decision Making (DODM)**: Service requests are the entities to make the offloading decision to minimized their own average response time, as in [27].
(3) **Application-aware Workload Allocation (AWA)**: As in [7], all the requests are assigned to the edge node with minimum transmission delay first. Then requests are reallocated to the edge node with minimum response time iteratively. However, there is no cooperation between edge nodes and the remote cloud server as service requests are not allowed to be further offloaded to remote cloud.

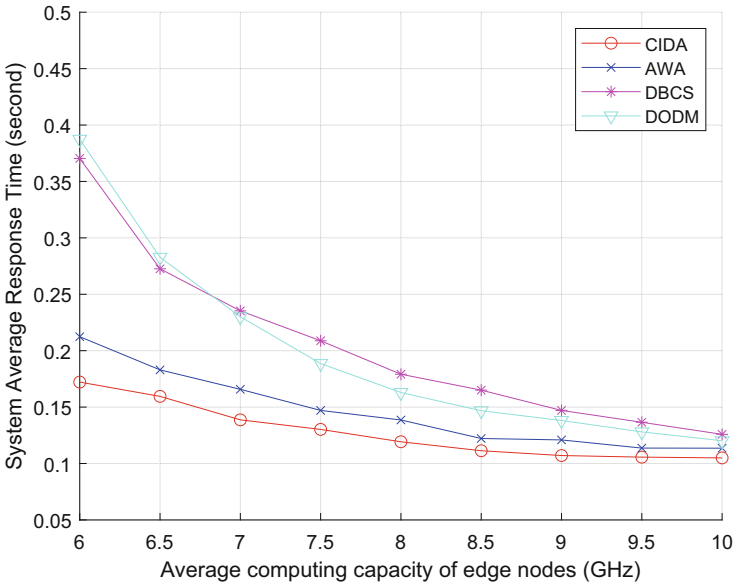

Fig. 3. Average response time over average computing capacity of edge nodes achieved by different algorithms

Figure 3 illustrates the system request average response time of the four algorithms mentioned above under different average computing capabilities of edge nodes (ranging from 6GHz to 10GHz). It can be observed that the system average response time of all algorithms exhibits a decreasing trend as the average computing capacity of edge nodes increases. When the average computing capability of edge nodes is 6GHz, CIDA reduces the system response time by 18.3%, 53.7%, and 55.8% compared to the AWA, DBCS, and DODM algorithms, respectively. As the capabilities of edge nodes increase, the performance gap between the other algorithms and CIDA gradually decreases. This is because the execution latency of system requests is no longer a performance bottleneck in the DBCS and DODM algorithms due to the enhanced computing capabilities of the edge

nodes. Relatively speaking, there is no significant reduction on the system average response time of CIDA and AWA algorithms, indicating that the execution latency of system requests has been effectively optimized.

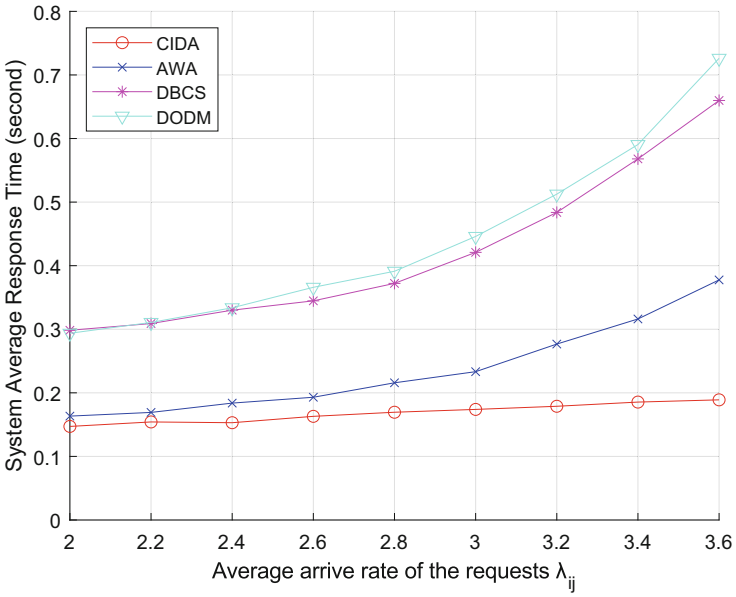

Fig. 4. Average response time over average request arrival rate λ_{ij} achieved by different algorithms

To investigate the performance of the above algorithms under different system workload level, we increased the average arrival rate of each type of request from 2 to 3.6. The experimental results are shown in Fig. 4. When $\lambda_{ij} = 2$, CIDA algorithm exhibits a significantly lower system average response time compared to the DBCS and DODM algorithms, while the system average response time of the AWA algorithm is relatively close to the CIDA algorithm. This result can be explained by the fact that in the DBCS algorithm, the assignment of system requests depends on the average workload level among edge nodes without considering the heterogeneity of them. In the DODM algorithm, individual offloading decisions are made by each system request based on its own average response time, resulting in the poor cooperation among edge nodes. These factors contribute to the inferior performance of the DBCS and DODM algorithms compared to the CIDA and AWA algorithms. However, as λ_{ij} increases, the performance gap between the other algorithms and the CIDA algorithm gradually widens. When λ equals to 3.6, CIDA algorithm reduces the system average response time by 49.7%, 71.4%, and 74.4% compared to AWA, DBCS and DODM algorithms, respectively. For AWA algorithm, since there is no cooperation between the cloud server and the edge nodes, the edge nodes are getting

heavy-loaded as the average request arrival rate increases, which leads to much higher execution delay for the requests. In contrast, CIDA can further offload a portion of the requests with high computation requirement to the remote cloud server, ensuring efficient system operation even under high average requests arrival rate.

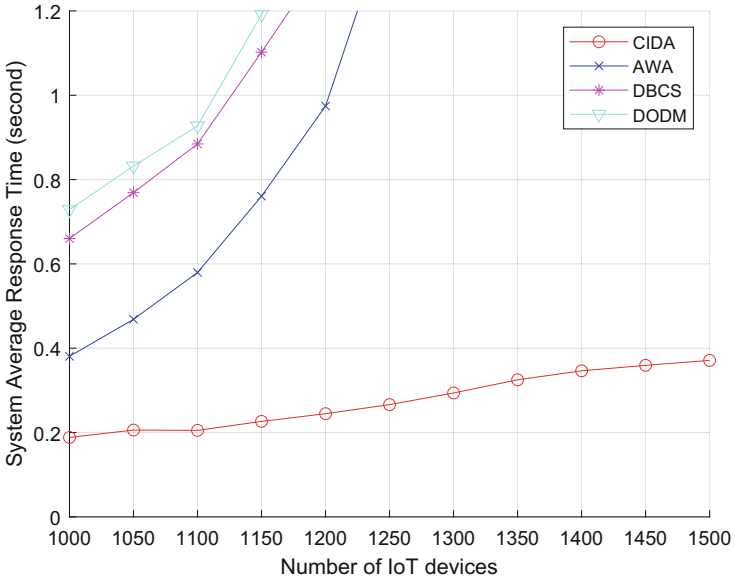

Fig. 5. Average response time over numbers of IoT devices achieved by different algorithms

To further validate the scalability of our algorithm, we increased the number of IoT devices from 1000 to 1500 while maintaining the average request arrival rate λ_{ij} at 3.6. The results are shown in Fig. 5. It can be observed that as the number of devices increases, the system average response time of AWA, DBCS, and DODM algorithms dramatically increases, and reach the system capacity limit as the number of devices approaches 1300. Relatively speaking, by leveraging the collaboration between edge nodes and the remote cloud server, CIDA exhibits only a slight increase in the system average response time, which clearly indicates the effectiveness and high scalability of CIDA.

7 Conclusions

In this paper, we systematically investigate the problem of service-aware cooperative task offloading and scheduling in a three-tier multi-access edge computing IoT environment. This problem involves the joint optimization of the system offloading profile and computing capacity allocation profile, which has been

proven to be at least NP-hard. To address this problem, we propose a two-layer heuristic algorithm CIDA, where the first layer obtain a suboptimal offloading profile with minimized response time for the service requests at a low time complexity, and the second layer allocate the computing capacity on the edge nodes to each type of services optimally. Simulation results demonstrate that CIDA outperforms other approaches in terms of minimizing the average response time for service requests. Additionally, CIDA exhibits high scalability as the system size increases, making it suitable for large-scale deployment.

Acknowledgements. This work was supported in part by the Guangdong Key Construction Discipline Research Ability Enhancement Project (Grant No. 2021ZDJS086); in part by the Guangdong University Key Project (Grant No. 2019KZDXM012); in part by the Natural Science Foundation of Guangdong Province (Grant No. 2021A1515010656); in part by Guangdong Basic and Applied Basic Research Foundation (2022B1515120059); in part by the research team project of Dongguan University of Technology (Grant No. TDY-B2019009).

References

1. Laghari, A.A., Wu, K., Laghari, R.A., Ali, M., Khan, A.A.: A review and state of art of internet of things (iot). Archives of Computational Methods in Engineering, pp. 1–19 (2021)
2. Tao, M., Li, X., Wei, W., Yuan, H.: Jointly optimization for activity recognition in secure iot-enabled elderly care applications. Appl. Soft Comput. **99**, 106788 (2021)
3. Zhu, R., Liu, L., Song, H., Ma, M.: Multi-access edge computing enabled internet of things: advances and novel applications (2020)
4. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutorials **19**(3), 1628–1656 (2017)
5. Xu, J., Chen, L., Zhou, P.: Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 207–215. IEEE (2018)
6. Ma, X., Zhou, A., Zhang, S., Wang, S.: Cooperative service caching and workload scheduling in mobile edge computing. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 2076–2085. IEEE (2020)
7. Fan, Q., Ansari, N.: Application aware workload allocation for edge computing-based iot. IEEE Internet Things J. **5**(3), 2146–2153 (2018)
8. Tao, M., Xueqiang, L., Kaoru, O., Mianxiong, D.: Single-cell multi-user computation offloading in dynamic pricing-aided mobile edge computing. IEEE Trans. Comput. Social Syst. (2023). https://doi.org/10.1109/TCSS.2023.3308563
9. Guo, F., Zhang, H., Ji, H., Li, X., Leung, V.C.: An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Trans. Network. **26**(6), 2651–2664 (2018)
10. Fan, Q., Ansari, N.: Towards workload balancing in fog computing empowered iot. IEEE Trans. Netw. Sci. Eng. **7**(1), 253–262 (2018)
11. Zhu, X., Zhou, M.: Multiobjective optimized cloudlet deployment and task offloading for mobile-edge computing. IEEE Internet Things J. **8**(20), 15582–15595 (2021)
12. Cui, Y., Song, J., Ren, K., Li, M., Li, Z., Ren, Q., Zhang, Y.: Software defined cooperative offloading for mobile cloudlets. IEEE/ACM Trans. Netw. **25**(3), 1746–1760 (2017)

13. Tao, M., Li, X., Yuan, H., Wei, W.: Uav-aided trustworthy data collection in federated-wsn-enabled iot applications. Inf. Sci. **532**, 155–169 (2020)
14. Tong, L., Li, Y., Gao, W.: A hierarchical edge cloud architecture for mobile computing. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9. IEEE (2016)
15. Tao, M., Ota, K., Dong, M.: Dsarp: dependable scheduling with active replica placement for workflow applications in cloud computing. IEEE Trans. Cloud Comput. **8**(4), 1069–1078 (2020)
16. Tan, H., Han, Z., Li, X.Y., Lau, F.C.: Online job dispatching and scheduling in edge-clouds. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1–9. IEEE (2017)
17. Chen, M.H., Dong, M., Liang, B.: Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints. IEEE Trans. Mob. Comput. **17**(12), 2868–2881 (2018)
18. Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, F.R., Han, Z.: Computing resource allocation in three-tier iot fog networks: a joint optimization approach combining stackelberg game and matching. IEEE Internet Things J. **4**(5), 1204–1215 (2017)
19. El Haber, E., Nguyen, T.M., Assi, C.: Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds. IEEE Trans. Commun. **67**(5), 3407–3421 (2019)
20. Chen, Z., He, L.: Modelling task offloading mobile edge computing. In: 2022 The 8th International Conference on Computing and Data Engineering, pp. 15–21 (2022)
21. Lyu, X., Tian, H., Sengul, C., Zhang, P.: Multiuser joint task offloading and resource optimization in proximate clouds. IEEE Trans. Veh. Technol. **66**(4), 3435–3447 (2016)
22. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun. **36**(3), 587–597 (2018)
23. Tao, M., Ota, K., Dong, M., Yuan, H.: Stackelberg game-based pricing and offloading in mobile edge computing. IEEE Wireless Commun. Lett. **11**(5), 883–887 (2022)
24. Van Adrichem, N.L., Doerr, C., Kuipers, F.A.: Opennetmon: network monitoring in openflow software-defined networks. In: 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1–8. IEEE (2014)
25. Sun, X., Ansari, N.: Latency aware workload offloading in the cloudlet network. IEEE Commun. Lett. **21**(7), 1481–1484 (2017)
26. Jia, M., Cao, J., Liang, W.: Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Trans. Cloud Comput. **5**(4), 725–737 (2015)
27. Gao, B., He, L., Jarvis, S.A.: Offload decision models and the price of anarchy in mobile cloud application ecosystems. IEEE Access **3**, 3125–3137 (2015)