Balanced Offloading of Multiple Task Types in Mobile Edge Computing

Ye Zhang, Xingyun He, Jin Xing, Wuyungerile Li**
School of Computer Science
Inner Mongolia University
Hohhot, China
**gerile@imu.edu.cn

Winston K.G. Seah
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
winston.seah@ecs.vuw.ac.nz

***Abstract*—The continuous development of mobile networks poses new challenges for end devices with limited computing power. Mobile or multi-access edge computing (MEC) has been proposed for providing the computing resources close to the end devices that need them. However, in real network environments, MEC servers have limited computing resources that need to be shared among many devices and efficient resource allocation is critical to ensure that the limited resources are optimally used. In view of this, we propose the Balanced Offload for Multi-type Tasks (BOMT) algorithm. The tasks to be offloaded are first prioritised according to their type, size and maximum tolerable delay; then different offloading algorithms are executed for different priority tasks according to the level of the priority and the current load on the MEC server. Following which, the optimal offloading policy is determined iteratively. Simulation results show that BOMT can effectively reduce system latency, increase user coverage and offload task completion rates.**

***Index Terms*—MEC, compute offload, load balancing, multi-class services.**

I. Introduction

The massive growth of 5th generation and beyond (5G/B5G) mobile networks has triggered a proliferation of Internet services and applications, which increases the demand for data storage and processing, posing even greater challenges to smart mobile devices. This motivated the development of fog computing (FC) and mobile edge computing (MEC) [1] to provide cloud computing resources close to the end devices. The concept of Mobile Edge Computing emerged in 2013 and was later officially launched by the European Telecommunications Standards Institute (ETSI) in 2014 [2]. The basic idea is to migrate the cloud computing resources from inside the mobile core network to the edge of the mobile access network, so that the network edge has the capability to process mobile terminal applications. That is, MEC technology distributes computing and storage resources near MTs, which can overcome their own resource shortage by transferring functions such as computing and storage to MEC servers.

As the variety of mobile devices increases, the types of mobile applications also increase. Furthermore, real network environments are often complex and diverse, with multiple types of tasks, such as voice calls, video streaming, and file transfers. Moreover, the distances between different tasks and their corresponding MEC servers are also different, giving rise to a range of tasks with diverse service requirements.

Therefore, we focus on the problem of offloading multiple types of tasks in mobile edge computing, and propose an efficient Balanced Offload for Multi-type Tasks (BOMT) algorithm to maximise the number of users served and to meet the needs of different users, ensuring that high priority tasks are able to reduce their task computation times, thus satisfying their latency requirements, and ensuring that low priority tasks also complete their computation within the latency allowed. Our key contributions are:

- A prioritisation scheme that takes into account factors, such as, task data size, maximum tolerated latency and task type;
- An iterative method that computes the offloading schedule based on the tasks' priority level and server loads.

In the next section, we outline the work related to load balancing and task prioritisation in MEC offloading. We then build a real network environment based on an LTE network with the aim of observing whether task type and size may have an impact on offloading decisions. With the data measurements in the real scenario, we can better understand the network architecture of mobile edge computing, and thus design simulation experiments that are closer to the real scenario to verify the superiority of the proposed algorithm. This is followed by a detailed discussion of our proposed BOMT algorithm and related concepts. Then, we validate our BOTM algorithm and compare it with different offloading algorithms. The analysis of the results shows that the BOMT algorithm can balance the load of MEC servers, improve user coverage, and reduce task execution latency. Finally, we conclude this paper and discuss future work.

II. Related Work

Till recently, most previous research on mobile edge computing stays in the network scenario of a single MEC server and one generic type of task. As the network size increases and the network environment becomes more complex, scenarios involving multiple MEC servers being deployed around users at the same time are increasingly common. Similarly, the variety

**Corresponding author

of tasks to be supported also increases. When MEC servers work independently, the state of other MEC servers is not considered, and there is no collaboration between the servers, which can result in some uneven distribution of resources. A single MEC server has limited resources, but MTs and the number of task requests they generate are exploding [3]. When multiple users make requests to the same MEC server concurrently, the server may be overloaded and cannot fulfill all users' requests.

Load balancing is used to dynamically adjust the load between MEC servers to avoid the above situation, maximise the use of MEC server resources, enable servers to compute more tasks, thus improving the overall system service performance, enhancing the QoS for users and reducing system latency. In line with the then prevailing trend, the concept of software-defined networking has been used for offloading of tasks in an ultra-dense network scenario with the aim of reducing latency while extending the battery life of MTs [4]. Collaborative information sharing between servers can help alleviate the problem of unbalanced server loading. Chen *et al*. [5] propose a new architecture for multiple MEC servers, called NeiMEC, which can improve the hit rate of content cache replacement by referring to the information of neighbouring MEC servers when the content cache of a MEC server is full. ECOP [6] is an energy-efficient computational offloading method which considers energy consumption, privacy protection and load balancing, by defining the problem as a multi-objective optimisation, and achieving optimisation of load balancing and energy consumption by means of an improved strength Pareto evolutionary algorithm.

As the traffic load increases, more effective ways to offload tasks to and among different servers are needed. Zhang *et al*. [7] propose a load balancing algorithm for redistributing tasks among small base stations, which offloads tasks from MTs to the best small base station based on the location of the user and the current number of users on the MEC server. Similarly, a game-based multitype task offloading scheme has been proposed for offloading tasks to MEC-enabled base stations [8].

The types of tasks that need to be offloaded in real life are diverse, and when computing and processing these tasks, it is not reasonable to allocate resources to tasks by relying only on information of where the task is located and the time requested to compute the offload [9]; the urgency of the task and the current load of the network should also be considered. Like the tasks, MEC servers vary in characteristics as well, such as being static or mobile [10]. One approach is to adaptively offload tasks based on their dwell time minimisation [11] by first dividing users into high-priority and low-priority users. The MEC server then uses a weighted polling scheduling strategy with tasks of high-priority users given more machine cycles than low-priority users, resulting in more efficient services. Similarly, an optimization scheme for resource allocation and task scheduling based on the urgency of task has been proposed [12] that aims to minimise the delay of urgent high-priority tasks and total system delay as a whole.

As expected, machine learning has been applied as the network scenario becomes increasingly more complex [13]. A multi-capability federated deep Q-network (M2FD) algorithm has proposed to optimize the objectives of saving time and energy in offloading tasks, as well as protect user data [14]. Another approach uses a deep reinforcement learning model based on the Actor-Critic algorithm to adaptively offload tasks to minimise the total penalty for time-bound and delay-sensitive tasks [15].

Nowadays, network environments are often complex and diverse, with multiple kinds of tasks, such as voice calls, video streaming and file transfers. It should also be noted that distance is an important factor in the performance of the system, with different tasks being at different distances from the MEC server and these different tasks having different requirements. In this paper, we propose an efficient offloading strategy to maximise the number of users served and to meet the needs of different users, ensuring that some high-priority tasks can be computed in less time while satisfying the latency, and that low-priority tasks can be computed without significant performance degradation.

III. LTE NETWORK-BASED DATA MEASUREMENT

To comprehensively understand the structural intricacies of mobile edge computing networks and model a more authentic operational environment conducive to the design of simulation experiments that closely mirror real-world scenarios, a practical mobile edge computing network testbed was established for empirical data collection. This was accomplished by utilizing LTE base stations and sub-networks thereof. In this study, an LTE-based mobile edge computing network testbed was employed, as it aligns more closely with actual operational circumstances. LTE, being a widely deployed mobile communication technology, closely emulates real network conditions. In the context of mobile edge computing research, given the prominence of existing network infrastructures, the adoption of LTE-based testbeds enables a more precise emulation of actual network conditions, encompassing signal propagation, network congestion, latency, and other essential attributes. Furthermore, LTE-based testbeds often adhere to standard LTE devices and protocols, which have undergone extensive validation and implementation, ensuring their efficacy in simulating genuine communication scenarios within controlled laboratory settings. This heightened level of confidence substantiates the reliability and replicability of experimental findings.

The network structure of the experimental testbed was designed for task migration between MEC servers, to separately measure the different sizes and types of task transmission delays in various traffic scenarios. Fig. 1 are physical images of the experimental equipment, and the equipment used in the testbed (Fig. 2) is shown in Table I.

The data types measured are file transfer, video and image packets, with file sizes in the range of 1$\sim$105MBytes, video sizes ranging from 5MBytes to 105MBytes and images ranging

(a) Mobile Device (b) Mobile WIFI Access Point

Fig. 1: Equipment used in experimental testbed

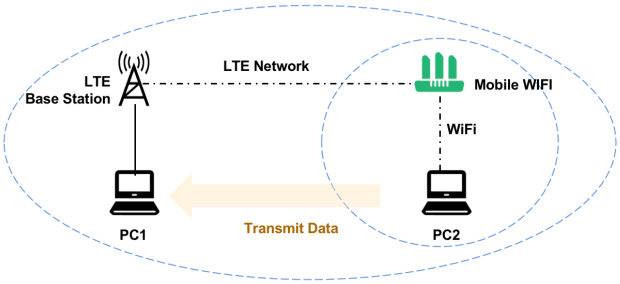


Fig. 2: Network Testbed

from 1MBytes to 20MBytes. The findings shown in Fig. 3 depict an observable correlation between the size of transmitted data and the corresponding latency, with this trend remaining consistent across the spectrum of data types. This uniformity in latency performance could potentially be attributed to the influence of queueing and buffering mechanisms intrinsic to network devices. These mechanisms contribute to latency irrespective of data size, as their behavior is contingent on variables such as network congestion, traffic prioritization, and offloading determinations. Notably, differences in latency between larger file transfers and video streaming can be ascribed to the distinctive underlying transport layer protocol – Transmission Control Protocol (TCP) for file transfers, which entails reliable packet delivery through retransmission of lost packets.

Analyzing the measurement results reveals that the transmission latency for diverse data types remains indistinguishable for data sizes up to approximately 40 MBytes. Beyond this threshold, the impact of the underlying transport protocols becomes more pronounced, particularly in scenarios employing TCP where congestion and flow control mechanisms elevate latency. Consequently, file transfers of sizes exceeding 40 MBytes exhibit higher latency than equivalently sized video streaming data using the User Datagram Protocol (UDP). Although cloud-based offloading holds a more dominant role, the impact of the distance between the MT and the MEC server on transmission delay should not be underestimated. Longer distances correspond to increased transmission delays.

To provide a comprehensive understanding of mobile edge computing across various scenarios, a network testbed was established to facilitate task migration between MEC servers in real-world field conditions. This setup allowed for the measurement of relevant data transfer latencies, contributing to a more insightful understanding of mobile edge computing dynamics. Notably, the collected data underscores the convergence of transmission latencies across diverse data types, despite variations in data sizes. This convergence suggests that factors beyond data size, such as task type, may influence offloading decisions. This observation sets the stage for the introduction of the Balanced Offloading for Multiple Task Types (BOMT) algorithm in the subsequent section.

TABLE I: Equipment used in Experimental Testbed

Type of equipment	Models
LTE Base Stations	ZXTD-LTE R8968 RF System
Mobile WIFI	BFI Series Portable LTE Router F7 MiFi
Mobile Devices	GH820
PC1 (MEC Server)	Dell Inspiron 5580
PC2 (MEC Server)	Asus VM590Z

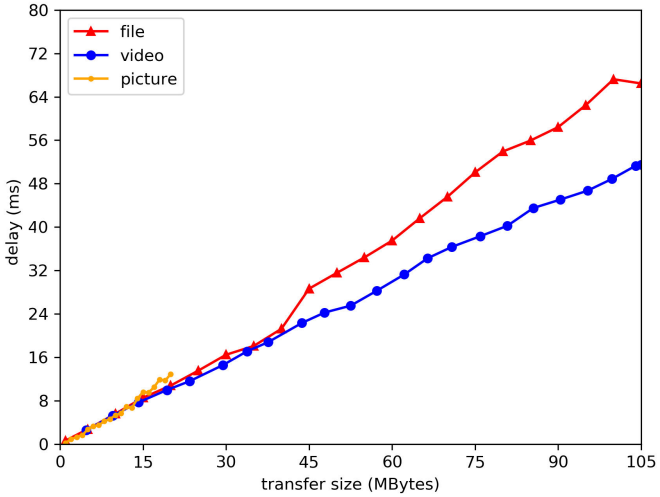


Fig. 3: Task migration latency between MEC servers

IV. Balanced Offload for Multi-type Tasks

In this section, we first describe the network model, the task offloading model and the problem description, and finally detail the BOMT algorithm.

A. Network Model

We consider a network consisting of macro base stations and small base stations, as shown in Fig. 4. The macro base station and small base stations are equipped with computing capability to form an MEC server with the base station, denoted by the set $X = \{X_1, X_2, ..., X_M\}$, each capable of executing multiple offloaded tasks simultaneously. For example, the service area of the macro base station X_a covers the service area of the smaller base stations X_c and X_d. In particular, the macro base station X_a's coverage area also overlaps with the coverage area of the other neighbouring macro base station X_b, and similarly all MEC servers have a certain overlaps in their service areas. When an MT is within the service area of an MEC server, it

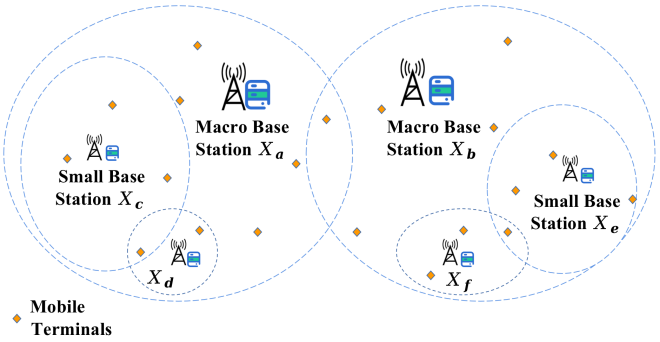


Fig. 4: Network example

can offload its computation tasks to the MEC server, which will perform the task and return the results to the MT.

It is assumed that MTs are randomly located within the coverage area of each MEC server and, for simplicity, each MT has only one task to be offloaded. The number of tasks in the entire network system varies with time, and tasks can be offloaded to any MEC server when the MT is located within the service area of the MEC server. Each MEC server has different service coverage radius and computing resources and the number of offloaded tasks, task types and data sizes distributed within the service range of the MEC server also varies with different quality of service (QoS) requirements. MTs fall within the service coverage of at least one MEC server, sometimes two or more, depending on their locations, and they offload their computing tasks to an appropriate MEC server via a wireless link.

Considering that the distance between the MT and the MEC server affects the transmission delay of the task, the uplink transmission rate of the wireless link for offloading task T_i to the MEC server M_j is shown in Eqn. (1) [16],

$$R_{ij} = B_{ij} \log_2 \left(1 + \frac{P_i h_{ij} ||p_{T_i} - p_{M_j}||^{-\theta}}{\sigma^2} \right) \tag{1}$$

where B_{ij} denotes the wireless bandwidth allocated for task T_i, P_i denotes the power used by task T_i to transmit data over the wireless link, h_{ij} denotes the channel gain between task T_i, $p_{T_i}$is the current location of task T_i, p_{M_j} is the current location of MEC server M_j, $|p_{T_i} - p_{M_j}|$ denotes the distance between task T_i and MEC server M_j, σ^2 denotes the noise power, and θ denotes the path loss index.

B. Task Offload Model

Suppose there are a total of N MTs randomly distributed in the network, and that each MT has only one task to be offloaded. Let $T = \{T_1, T_2, ..., T_i, ..., T_N\}$ denote the set of tasks, Q_i^c denote the CPU resources required for task T_i, $t_{i\_\max}$ denote the maximum latency tolerated by task T_i and S_i denote the data size of task T_i. As there are different task types running on the MTs, q is used to indicate the task type of the MT. For the purpose of illustration, we assume $q \in \{1, 2, 3\}$ is set to 1 for videos, 2 for pictures and 3 for files, indicating high, medium and low latency requirements respectively.

When an MT intends to offload a task, it needs to send an offload request containing basic information about the task to the base station first, and the user can only offload the task to the MEC server if executing the task at the MEC server incurs lower processing delay than executing the task locally. When the computational task is processed locally on the user's device, the processing delay includes only the computational latency of the task, t_i^l, and is calculated as shown in Eqn. (2),

$$t_i^l = \frac{Q_i^c}{f_i} \tag{2}$$

where f_i denotes the computational power of the end device containing the task T_i.

When the computation task is processed at the MEC server, the processing delay consists of three components, which are the latency of transmitting the task from the MT to the MEC server, the computation latency of the task and the transmission latency when the result of the task is returned. The user offloads the task via the wireless link and the data transmission latency, t_i^c, is calculated as shown in Eqn. (3),

$$t_i^c = \frac{S_i}{R_{ij}} \tag{3}$$

where S_i indicates the data size of the task T_i and R_{ij} is the transfer rate of the task T_i to the MEC server M_j. As can be seen from Eqn. (3), the transmission delay of the data is directly related to the size of the data. The returned result of the task is much smaller than the task, so we assume the transmission delay of the returned result to be negligible.

When the task T_i is offloaded to the MEC server M_j, the computational resources of the MEC server need to be allocated to the task T_i and the computational latency, t_{ij}^k, of the task is calculated as shown in Eqn. (4),

$$t_{ij}^k = \frac{Q_i^c}{f_{ij}} \tag{4}$$

where f_{ij} denotes the computing resources allocated to the task M_j by the MEC serve M_j. Therefore, the total processing delay for task T_i is calculated as shown below:

$$t_i = t_i^c + t_{ij}^k. \tag{5}$$

Considering that different types of tasks are offloaded, having different QoS requirements, these tasks need to be differentiated and prioritized in order to ensure high reliability and low latency services can be provided to some specific high-priority tasks while service levels that meet the basic needs are provided to the rest. To achieve this, we consider the priority of tasks in the offloading process, which is computed as shown in Eqn. (6).

$$\Pr = \frac{S_i}{t_{i\_\max} \times q} \tag{6}$$

where $t_{i\_\max}$ denotes the maximum tolerable delay of task T_i. In Eqn. (6), it can be seen that the larger the size of

data transferred by task T_i, the smaller the maximum tolerable delay, and the smaller the q value, the larger the Pr value of task priority, indicating the higher priority of task T_i.

This model evaluates the processing latency when tasks are executed locally and offloaded to the MEC server for execution. This includes consideration of factors such as network transmission latency, the computational power of the MEC server, and the computational power of the local terminal. By comparing the latency of the two execution methods, the model can select the optimal execution strategy. The model also considers the urgency and importance of the tasks and offloads high-priority tasks to the MEC server for execution first to ensure that user requirements are met in a timely manner. Most importantly, the model monitors the load of the MEC servers and selects idle or lightly loaded servers to execute the offloaded tasks. This helps to avoid overloading the servers and thus ensures that the tasks are processed in a shorter period of time.

C. Description of the problem

MTs are expected to be selfish by nature, i.e., they want their tasks to be computed as fast as possible. However, in a complex network, the uneven distribution of MTs and MEC servers and the concentration of users in hotspot areas make multiple MTs offload tasks to the same MEC server, which causes the MEC server to be overloaded, which in turn leads to an increase in the computation latency of the tasks, and reduces the user experience. Since MTs offload different types of tasks, and the size and maximum tolerable delay of each task are different, the quality of service requirements of tasks for MEC servers are also different, and different services should be provided for different tasks. A reasonable offloading strategy will improve the congestion of tasks on the MEC server and provide corresponding services according to different tasks.

Maximising the number of users served, and thus ensuring system load balancing, while providing different services for different types of offload tasks at a given time is the problem to be solved in this section. The resulting optimisation problem is as follows:

$$\begin{aligned} \max \quad & \sum_{t} \sum_{i=1}^{N} \sum_{j=1}^{M} z_{i,j}^{t} \\ s.t. \quad & i \in [1, N] \\ & j \in [1, M] \\ & \sum_{i}^{N^{'}} \varphi_{ij} < \varphi_{j,Max} \end{aligned} .$$

The objective of the optimisation problem is to maximise the number of users served $z_{i,j}^{t}$ per time slot t. The constraint on the number of users is N. The maximum number of MEC servers in the current network is M and the number of users that can be served by the MEC server is limited, while the number of tasks calculated is less than the maximum number of tasks that can be served by the MEC server, $N^{'}$ the number of users being served by the MEC server, φ_{ij} is M_j server's load due to T_i and $\varphi_{j,Max}$ is the maximum load that M_j can handle.

D. Algorithm Description

Given M MEC servers amd N MT, where each MT has a task to be offloaded, the set of tasks is denoted as $T = \{T_1, T_2, ..., T_i, ..., T_N\}$. We assume that the MT is stationary in the network for a short period of time when the offload algorithm executes and its location does not change.

After the MEC server receives the information about the tasks to be offloaded by the MT, it calculates the priority of each task scheduled to be offloaded according to the Eqn. (6) and sorts them in descending order, and classifies the tasks into three levels according to their sizes, with the priority of Pr1 being the highest, that of Pr2 being in the middle, and that of Pr3 being the lowest.

The steps of the algorithm are as follows:

1) First initialise the MEC server and the information about individual tasks;
2) Construct the set of tasks that need to be offloaded to the MEC server;
3) Sort the tasks in each priority class in descending order according to the priority value computed using Eqn (6);
4) Execute three different offloading algorithms (A, B and C below) for tasks of different priorities according to a certain ratio.

1) Algorithm A: Offload priority Pr1 tasks, i.e. $q = 1$, by finding the MEC server M_j^S that can process T_i with the shortest delay, according to Eqn. (5). If the current load of MEC server M_j^S is less than 80% of its maximum load, offload T_i to M_j^S; otherwise, find the server M_j^M with the next shortest processing latency for the task, and if the current load of MEC server M_j^M is less than 50% of its maximum load, offload the task T_i to MEC server M_j^M; if T_i is still not offloaded at this point, it must be offloaded to MEC server M_j^S in order to ensure the quality of Pr1 task completion.

When the network traffic reaches a peak of 80%, the network will be in a high load state, and at this time, the response speed of the system will be significantly reduced. In this case, if other traffic comes in again, the network will be very congested, which may lead to transmission interruption or failure. Choosing a load threshold of 50% can also prevent servers from overusing resources, thereby avoiding affecting the normal operation of other servers due to excessive load on some servers.

2) Algorithm B: Offload priority Pr2 tasks, i.e. $q = 2$; find the MEC server M_j^S that can process T_i with the shortest delay, according to Eqn. (5). If the current load of MEC server M_j^S is less than 60% of its maximum load, offload; otherwise, find the server M_j^M with the next shortest processing latency for the task, and if the offloading of task T_i to the MEC server M_j^M does not exceed its maximum load, offload it; if it is still not possible to offload at this point, in order to ensure the QoS of the Pr2 task and the tasks already offloaded on the MEC

server M_j^S, reassign the lower priority task $T_i^{'}$ on server M_j^M and then offload task T_i to server M_j^M.

In order to maintain the resource surplus of the server, so as to accommodate more tasks and ensure the stability and performance of the system when the task load increases, the MEC server load is selected to be less than 60% of its maximum load.

3) Algorithm C: Offload priority Pr3 tasks, i.e. $q = 3$, locating the MEC server M_j^M with the second shortest processing delay, according to Eqn. (5). If the current load of MEC server M_j^M is less than 60% of its maximum load, offload the task; otherwise, find the server M_j^L with the longest processing delay for task T_i. Assess the current load of server M_j^L, and if task T_i plus the current load of MEC server M_j^L is less than the maximum load of server M_j^L then offload, otherwise enqueue the task for the next time slot.

Considering that different types of tasks have different requirements on network service quality, the BOMT algorithm divides tasks into different priority levels according to task type, task size and maximum tolerable delay of tasks. We propose different offloading algorithms for tasks of different priorities to provide services that meet the requirements of different types of tasks. When the MEC server is full, the lower priority tasks are handed over to other free MEC servers for computation, thus providing better service to the higher priority tasks. This improves resource utilization and achieves a balanced load on the MEC servers.

V. Validation And Performance

A. Simulation Environment and Parameter Settings

The MEC network was set up in a 200m diameter area, with MEC server M_1 located in position (0,0) in the coordinate system and a service coverage radius of 100m; server M_2's position is (-25,0) in the coordinate system with a service coverage radius of 75m; and M_3's position is (0,50) in the coordinate system and a service coverage radius of 25m. In each time slot, several MTs enter the area at random and remain in a stationary state.

Through simulation experiments, our BOMT algorithm is compared with random offloading, nearest server offloading, load balancing (LB) algorithm [7] and time-sensitive multi-user (TSMU) algorithm [17] in terms of system execution delay, task coverage, task completion rate and MEC server load.

1) Random offloading: the task is offloaded to a randomly selected MEC server whose signal range covers the task and is not fully loaded.
2) Nearest server: offloads a task to the closest MEC server that has coverage over the task and is not fully loaded.
3) LB algorithm: after offloading a task to an MEC server, the task that is at the junction of the signal coverage of multiple MEC servers is reassigned to the best available MEC server.
4) TSMU algorithm: tasks to be offloaded are ordered according to their priority, and the task with the highest priority is offloaded to the MEC server with the highest computing power, whose signal range covers the task and is not fully loaded.

In order to study the effect of task offloading for several algorithms, the resources allocated for task offloading are set to a fixed value. The simulation parameters for this paper are shown in Table I below.

TABLE II: Simulation Parameters

Description of parameters	Parameter values
MEC servers number M	3
Sub-channel bandwidth B_{ij}	0.8MHz
Wireless channel transmission power P_i	1W
Noise power σ^2	2×10^{-13} W/Hz
Data task size d_{ta}	1M 10M
Number of CPU cycles Q_i^c	0.1GHz 2GHz
M_1 Computing power Q_1^m	50GHz
M_2 Computing power Q_2^m	30GHz
M_3 Computing power Q_3^m	10GHz
Computing resources f_{ij} allocated to tasks T_i	2.5GHz,2GHz,1GHz
Mobile device computing power f_i	0.5GHz
Channel power gain h_i	exp(1)
Maximum tolerable delay $t_{i\_\max}$ for tasks	1∼4s

B. Simulation Results and Analysis

1) System Execution Time Delay: The four algorithms and the BOMT algorithm are compared separately in terms of the total delay in completing all tasks for a number of tasks ranging from 45 to 225.

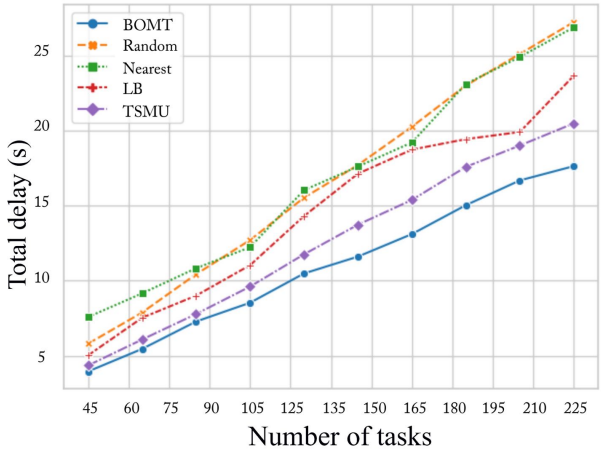


Fig. 5: Task execution times

The experimental results are shown in Fig. 5. When the number of tasks is small, the MEC server is less likely to be overloaded and the difference in total latency between the algorithms is not significant. As the number of tasks increases, the total latency to complete all tasks also increases.

With the same number of tasks, BOMT has lower system latency compared to all algorithms. This is because BOMT takes into account the task priority and the load of all servers in the network, effectively reducing the total delay of all task execution by making full use of the resources of the MEC

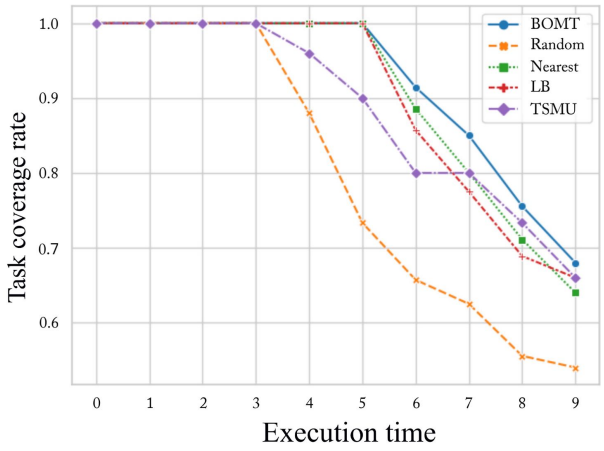


Fig. 6: Task coverage

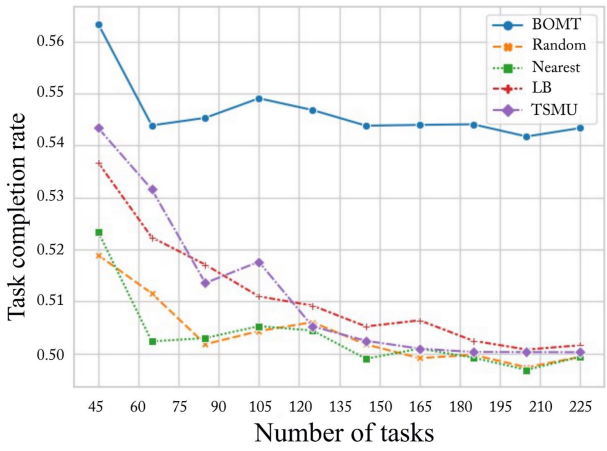


Fig. 7: Task completion rate

servers. The Random and Nearest algorithms do not consider load balancing in the MEC network and cannot cope with overloaded / underloaded MEC servers. The LB algorithm only considers the load situation of the MEC servers while TSMU only prioritises high priority tasks. Both LB and TSMU algorithms only consider the server with the shortest execution time when unloading tasks, ignoring other servers.

2) Task Coverage: Fig. 6 shows the number of tasks executed by the MEC servers as a proportion of the total number of tasks, which is set at 255.

In the first few seconds when the algorithms start executing, the coverage of several algorithms is relatively high as the number of tasks is small and has not yet reached the maximum number of tasks that can be served simultaneously within the MEC system. As time increases, the total number of tasks within the system increases, which can lead to individual MEC servers being overloaded and unable to service some of the newly arriving tasks within the service range. The most severe drop in task coverage was seen with the Random algorithm. For newly arriving tasks, the BOMT algorithm offloads according to their priority and dynamically adjusts for fully loaded MEC servers. As a result the BOMT algorithm is able to service more tasks and maintain a higher task coverage when the number of tasks is high.

3) Task Completion Rate: The task completion rate is the proportion of tasks that can be completed within the maximum tolerable time delay for the user to the total number of tasks offloaded.

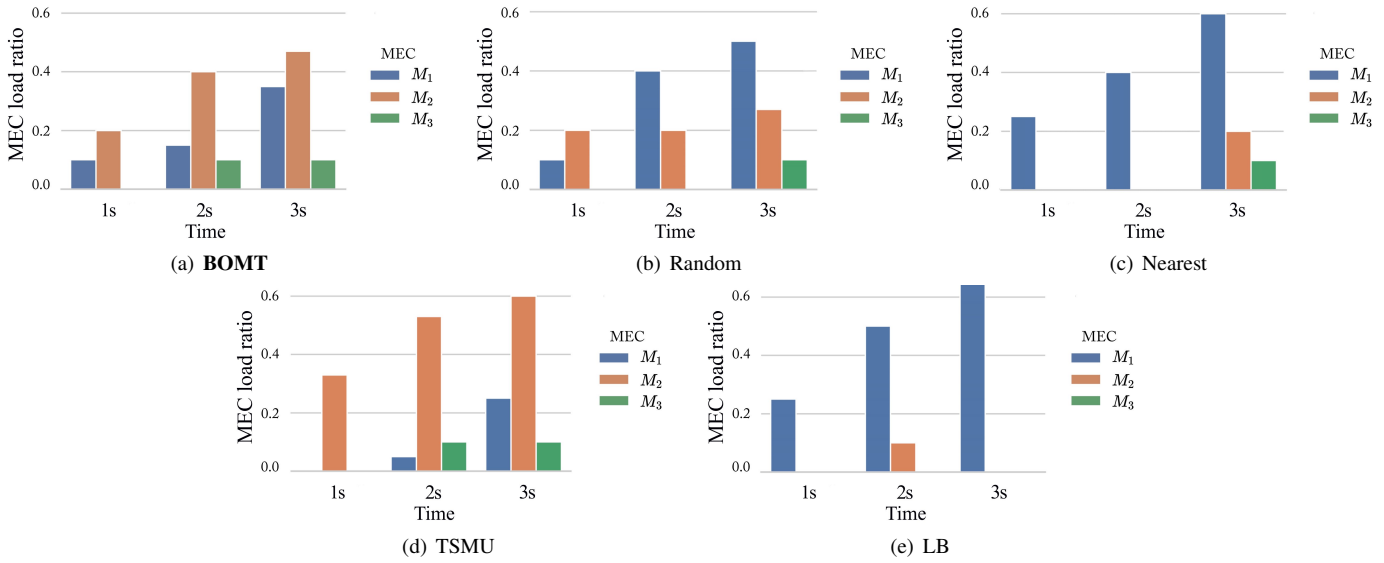
As shown in Fig. 7, when the number of tasks is small, the system can handle a certain amount of tasks, so the completion rates of several algorithms do not differ much per second, but as the number of tasks increases, multiple tasks arrive at each time slot. The comparison algorithms only consider servers where current tasks can be offloaded, while the BOMT algorithm prioritises offloading for newly arrived high priority tasks and reallocates tasks with lower priority and higher tolerated latency, making full use of the computing resources of all MEC servers in the system to serve more tasks while being able to maintain a high task completion rate.

4) MEC Server Load Profile: This section compares the load on each MEC server for different offloading algorithms. This is done by looking at the load ratio of each MEC for a randomly selected period of three consecutive seconds while the tasks are being offloaded. The load ratio of a MEC system represents the ratio of the number of tasks being served by each MEC server to the maximum number of tasks that can be served. Sometimes we may be more concerned about the performance of the system under transient load than the average performance of the whole process. The response of the system to a transient task or event can be evaluated by randomising the results for three consecutive seconds. The number of tasks is set to 100 for simplicity.

As shown in Fig. 8, for MEC server M_1, although most of the comparison algorithms result in a high load factor, for all the offloading algorithms, the Random offloading algorithm under utilises the computational resources of MEC server M_2. Both the LB and Nearest offloading algorithms over utilise the computational resources of MEC server M_1, which results in under utilisation of the other MEC servers and wastes computational resources. Conversely, the TSMU offloading algorithm overuses MEC server M_2, which may lead to overloading of M_2 while under utilising other MEC servers. The BOMT algorithm takes a holistic approach based on the differences in the priorities of the offloaded tasks and the performance of all the MEC servers in the system. The BOMT algorithm makes better (but not over) use of the most powerful MEC server M_1 and also utilises the other MEC servers M_2 and M_3 more to provide better service for all offloading tasks as a whole.

VI. Conclusions

In this work, we proposed a balanced offloading algorithm based on multiple types of tasks, referred to as Balanced Offload for Multi-type Tasks (BOMT), in mobile edge computing that considers multiple types of task, where different tasks have different requirements for quality of service. Tasks are prioritised based on their type, task data size and maximum



(a) **BOMT** (b) Random (c) Nearest

(d) TSMU (e) LB

Fig. 8: MEC server load profile

tolerable delay, and tasks with different priorities are offloaded to MEC servers with different service capabilities to provide services that meet their requirements, balancing the load on MEC servers, thereby improving system resource utilisation and enhancing user experience. Future work will consider user mobility, i.e., the impact of changes in the location of MTs on caching policies and offloading policies. More in-depth research can also be carried out on the MT itself, and the randomness and variability of the wireless channel, among other aspects.

Acknowledgement

This paper was supported by the Natural Science Foundation of Inner Mongolia Autonomous Region 2021MS06003.

References

[1] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proceedings of the Global Internet of Things Summit (GIoTS)*, Geneva, Switzerland, 6-9 June 2017.

[2] F. Giust, X. Costa-Perez, and A. Reznik, "Multi-Access Edge Computing: An Overview of ETSI MEC ISG," *IEEE 5G Tech Focus*, vol. 1, no. 4, December 2017.

[3] P. Jonsson, Ed., *Ericsson Mobility Report*. Ericsson, June 2023.

[4] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[5] Y.-C. Chen, C.-C. Wang, and J.-C. Chen, "NeiMEC: Automatically building neighbor relationship between mobile edge platforms in multi-access edge computing environment," in *Proceedings of the Third International Conference on Fog and Mobile Edge Computing (FMEC)*, Barcelona, Spain, 2018, pp. 20–25.

[6] X. Liu, X. Xu, Y. Yuan, X. Zhang, and W. Dou, "Energy-Efficient Computation Offloading with Privacy Preservation for Edge Computing-Enabled 5G Networks," in *Proceedings of the International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Atlanta, GA, USA, 14-17 July 2019, pp. 176–181.

[7] W.-Z. Zhang, I. A. Elgendy, M. Hammad, A. M. Iliyasu, X. Du, M. Guizani, and A. A. A. El-Latif, "Secure and Optimized Load Balancing for Multitier IoT and Edge-Cloud Computing Systems," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8119–8132, 2021.

[8] W. Fan, L. Yao, J. Han, F. Wu, and Y. Liu, "Game-based multitype task offloading among mobile-edge-computing-enabled base stations," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 691–17 704, 2021.

[9] D. Song, L. Rui, S. Chen, and X. Qiu, "A Computational Offloading Method Based on Resource Joint Optimization," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Chengdu, China, 4-6 August 2021, pp. 1–7.

[10] R. Zhang, L. Wu, S. Cao, X. Hu, S. Xue, D. Wu, and Q. Li, "Task Offloading with Task Classification and Offloading Nodes Selection for MEC-Enabled IoV," *ACM Transactions on Internet Technology*, vol. 22, no. 2, oct 2021.

[11] K. Aliobory and M. A. Yazici, "An adaptive offloading decision scheme in two-class mobile edge computing systems," in *Proceedings of the 41st International Conference on Telecommunications and Signal Processing (TSP)*, Athens, Greece, 4-6 July 2018, pp. 1–5.

[12] J. X. Liao and X. W. Wu, "Resource Allocation and Task Scheduling Scheme in Priority-Based Hierarchical Edge Computing System," in *19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Xuzhou, China, 16-19 October 2020, pp. 46–49.

[13] X. Zhang and S. Debroy, "Resource management in mobile edge computing: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 13s, jul 2023.

[14] Z. Tong, J. Wang, J. Mei, K. Li, W. Li, and K. Li, "Multi-type task offloading for wireless Internet of Things by federated deep reinforcement learning," *Future Generation Computer Systems*, vol. 145, pp. 536–549, 2023.

[15] T. Zhang, Y.-H. Chiang, C. Borcea, and Y. Ji, "Learning-Based Offloading of Tasks with Diverse Delay Sensitivities for Mobile Edge Computing," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 9-13 December 2019, pp. 1–6.

[16] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Joint computation offloading, resource allocation and content caching in cellular networks with mobile edge computing," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Paris, France, 21-25 May 2017, pp. 1–6.

[17] J. Zhang, B. Jiang, H. Zhao, and Y. Xu, "Time-Sensitive Multi-User Oriented Mobile Edge Computing Task Scheduling Algorithm," in *Proceedings of the 2nd International Conference on Computer Communication and the Internet (ICCCI)*, Nagoya, Japan, 26-29 June 2020, pp. 145–149.