applied sciences

MDPI

Article

Joint Computation Offloading and Data Caching Based on Cooperation of Mobile-Edge-Computing-Enabled Base Stations

Tian Liu [1], Wenhao Fan [2,*], Fan Wu [2], Wei Xie [1] and Wen Yuan [2]

[1] Southwest China Institute of Electronic Technology, Chengdu 610036, China; Ti_Liu@126.com (T.L.); raysheik@foxmail.com (W.X.)
[2] Beijing Key Laboratory of Work Safety Intelligent Monitoring, School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; wufanwww@bupt.edu.cn (F.W.); laurasugar@sina.com (W.Y.)
* Correspondence: whfan@bupt.edu.cn

Abstract: Mobile terminal applications with high computing complexity and high time delay sensitivity are developing quite fast today, which aggravates the load of mobile cloud computing and storage and further leads to network congestion and service quality decline. Mobile edge computing (MEC) is a way of breaking through the limits of computing and storage resources of mobile cloud and alleviating the load of mobile cloud. Computing time costs and transmission time costs are considered to be the main issues for the mobile cloud when carrying out computing offloading and data caching. Therefore, an efficient resource management strategy, which could minimize the system delay, is proposed in this paper. The new scheme offloads reasonably computing tasks and caches the tasks' data from the mobile cloud to mobile edge computing-enabled base stations. An intelligence algorithm, genetic algorithm, is being used to solve the global optimization problem which would cause transmission delay and computing resources occupation, and to determine the computing offloading and data caching probability. The simulation of the system using MATLAB is conducted in 8 different scenarios with different parameters. The results show that our new scheme improves the system computing speed and optimizes the user experience in all scenarios, compared with the scheme without data caching and the scheme without computing offloading and data caching.

Keywords: mobile edge computing; resource allocation; computing offloading; data caching; optimization

Citation: Liu, T.; Fan, W.; Wu, F.; Xie, W.; Yuan, W. Joint Computation Offloading and Data Caching Based on Cooperation of Mobile-Edge-Computing-Enabled Base Stations. *Appl. Sci.* **2021**, *11*, 5802. https://doi.org/10.3390/app11135802

Academic Editor: Luis Javier Garcia Villalba

Received: 24 May 2021
Accepted: 18 June 2021
Published: 22 June 2021

1. Introduction

With the rapid development of the mobile Internet and the Internet of things in recent years, the functions of the mobile terminals (MTs) are becoming much more rich than ever before. The character of mobile terminals has gradually evolved from a simple communication tool to a powerful station integrating communication, computing, entertainment and office. Various applications, such as augmented reality, virtual reality, and location-based service (LBS), have been contained in one mobile terminal as required by the consumers. These typical applications with high computing complexity and long time delay sensitivity not only aggravate the load of the mobile cloud (C) in computing and storage resources, but they also lead to system network congestion and service quality decline.

In order to alleviate the computing load of the mobile cloud, the concept of Mobile Edge Computation (MEC) is proposed [1], which provides the IT service environment and the cloud computing capability on the mobile network edge [2]. By deploying the mobile edge computing-enabled base station (MEC-BS) of MTs in a community and the neighbor mobile edge computing-enabled base stations of the MEC-BS (MEC-NBS) on the edge of the mobile network, the computing can sink to the mobile edge node, which can effectively reduce the load of the mobile cloud and reduce the demand for the data transmission bandwidth.

In our research, we define the MEC stations consists of five units: (1) receiving unit: which is used to receive service requests from its covering MTs and the surrounding MEC stations; (2) control unit: which is used to determine whether the received task is further offloaded and whether cache the data required for the corresponding task from C based on the Cooperative Resource Management Algorithm; (3) caching unit: which is used to cache the data required for the corresponding task based on the Cooperative Resource Management Algorithm to reduce the time delay in data access to C; (4) computing unit: which is used to calculate the computing tasks offloading from its covering MTs; (5) sending unit: which is used to send the calculation result to the MT, send the offloading request to further MEC stations and send the data request of the corresponding task to C. It is worth noting that MEC stations have limited computation and storage resources, so that they cannot provide computing and caching services for all the tasks like C. However, the MEC-BS will be still overloaded if there are too many tasks offloaded from the MTs. In existing algorithms, they try to ease the load of MEC stations through refusing, delaying or queuing the offloading requests of MTs. However, these algorithms will lead to poor QoS of the system. Thus, a new model of resource management is urgently needed.

In this paper, our research is concerned with a local system under mobile cloud, which includes a mobile edge computing-enabled base station (MEC-BS), the mobile terminals covered by this MEC-BS, and the neighbor mobile edge computing-enabled base stations within a certain distance from MEC-BS. Our goal is to offload the mobile cloud computing and storage pressure through MEC stations. Because of the poor computing capability of every MTs, we ignore MT as a offloading object to guarantee the quality of service. All tasks of MT need to go through the MT's MEC-BS to offload to mobile cloud or any neighbor mobile edge computing-enabled base station. Figure 1 is the overall architecture of our model.

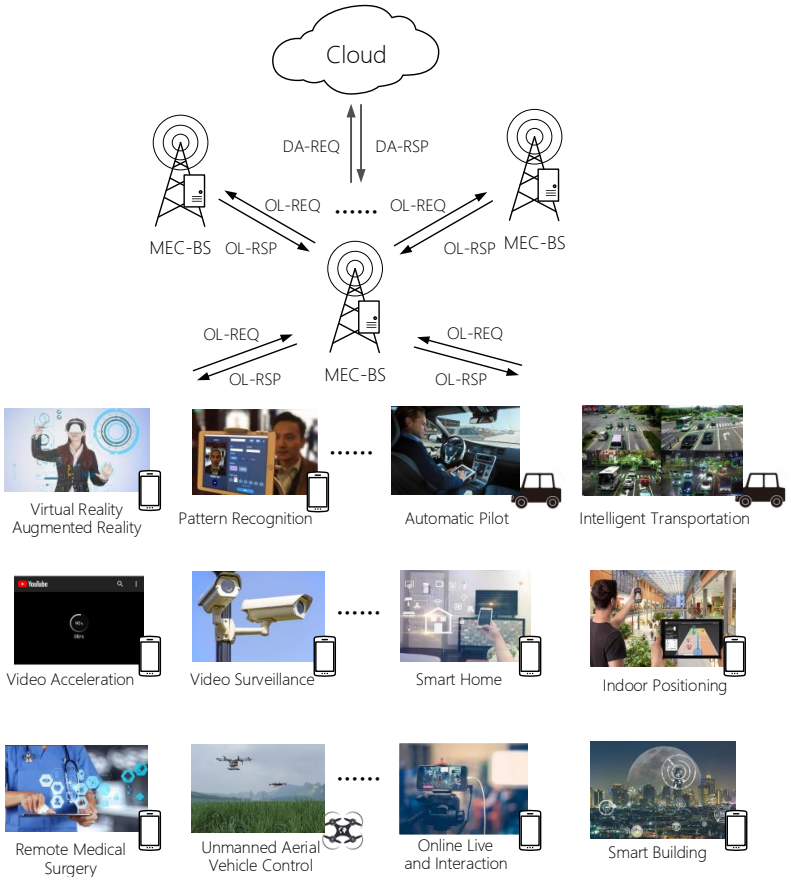

Figure 1. Offloading to MEC-BS, offloading to MEC-NBS, and executing in C.

As shown in Figure 1, the mobile edge computing can enhance the performance of multiple computation-intensive and delay-sensitive applications, such as virtual reality, augmented reality, pattern recognition, automatic pilot, intelligent transportation, video acceleration, video surveillance, smart home, indoor positioning, remote medical surgery, unmanned aerial vehicle control, online live and interaction, smart building, etc. The users can experience higher real-time performance of applications, and run more complex applications in their resource-constrained MTs. The steps for computing offloading and data caching under the mobile edge computing environments are as follows: (1) when MT has a new computing task, it will upload the offloading request (OL-REQ) to its MEC-BS; (2) if the task is determined to be executed in MEC-BS by the Cooperative Resource Management Algorithm and MEC-BS has cached data of the task, the task will be executed directly in MEC-BS, then the offloading response (OL-RSP) will be returned to MT; (3) if the task is determined to be executed in MEC-BS and there is no cached data in it, MEC-BS will send the data request (DA-REQ) of the task to C, then C will return the data response (DA-RSP) to MEC-BS. The next process of executing and delivering is the same as 2); (4) if the task is decided to be executed in MEC-NBS by the algorithm, MEC-BS will send the offloading request to a MEC-NBS. If MEC-NBS has cached the computing data before, it will execute the task and return the offloading response to MEC-BS, and then MEC-BS will return the offloading response to MT; (5) if MEC-NBS is decided to execute the task and it has not cached the computing data of the task in advance, it will send the data request to C, then C will return the data response to it. The next process of executing and delivering is the same as (4); (6) if the task is resolved to be offloaded to C by the algorithm, MEC-BS will send the offloading request to C, then C will execute the task and return the offloading response to MEC-BS, and then the offloading response will be returned to MT.

Our research work can be described briefly by a block diagram, as shown in Figure 2.

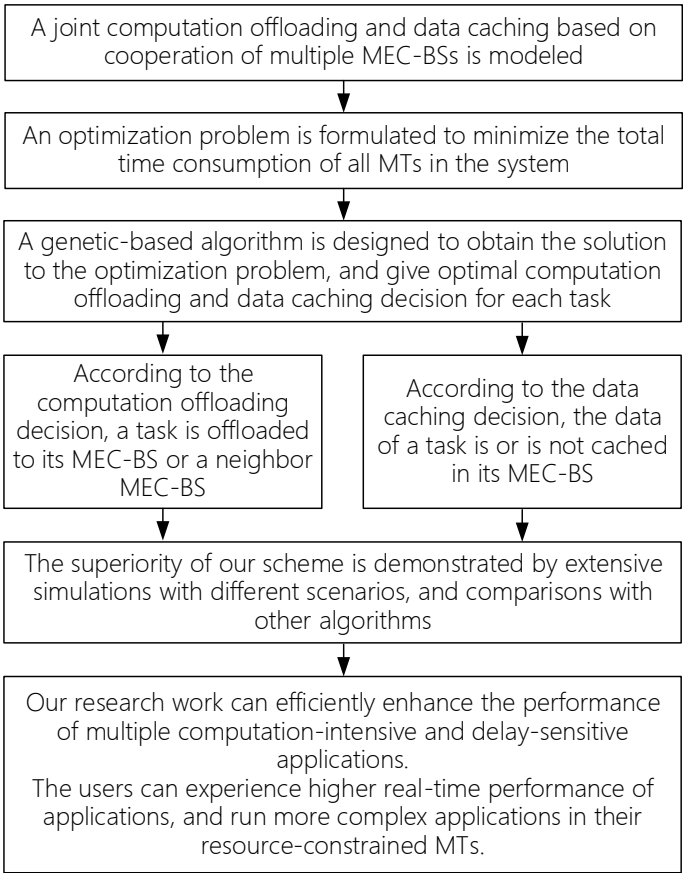

Figure 2. Brief introduction of the research work in our paper.

2. Related Works

The research work of MEC mainly focuses on the research of computing offloading system. It can be divided into architecture design, protocol design, resource management, and application in concept. It involves many fields, such as computing, communication, and security. The accumulated research results laid a foundation for the engineering implementation and theoretical system of mobile edge computing [3–6]. Resource management technology is one of the core technologies of mobile edge computing. It optimally uses system resources to optimize system performance. At present, the existing research work and main problems of resource management technology in MEC are as follows: A. computing resources and storage resources are independently disposed. Existing joint resource management only considers computing and wireless resources; B. the base station resources are optimized separately, lack of a unified multi base station collaboration mode and the corresponding resource management technology; C. existing offloading/caching decision algorithms lack algorithm cost optimization, high complexity and poor real-time performance, that affect the real-time performance of decision execution; D. lack of resource management technology for cloud computing server load and network load optimization.

2.1. Existing Computing Resources and Storage Resources

In the existing research based on mobile communication network scenarios, most of the research work only takes into accounting the problem of computing offloading (such as References [7,8]), or data caching (such as References [9,10]). In joint resource management, computing offloading is only combined with radio resource optimization (such as References [11–14]). For example, in the literature [11], the author proposes an effective radio and computing resource allocation scheme to minimize the total processing time of multiple tasks. Considering the local processing, each mobile user divides the computing task into unloading task and local task, which realizes the effective utilization of wireless power resources and computing resources. In the literature [12], the cache decision and radio and computing resource allocation of video service in mobile edge computing (MEC) are optimized to maximize the system revenue. In order to tolerate the uncertainty of network traffic and avoid the "hard constraint" based on constant content request rate, this paper uses robust optimization to obtain the optimal cache decision, and then allocates radio and computing resources for video trans-coding. In Reference [13], an online combined wireless and computing resource management algorithm for multi-user mobile edge computing system is proposed, which reduces the terminal energy consumption by assigning the CPU frequency, transmitting power and bandwidth of the mobile terminal.

In most mobile applications, the computing and storage requirements of the terminal are tightly coupled. That is, both of the computing process work and the data access work are included in one task, and the two are interrelated. Therefore, the joint optimization of computing and storage resources needs to be further explored in the research of resource management technology.

2.2. The Base Station Resources Are Optimized Separately

In the existing research work, most of the documents are based on the mobile edge of single base station to compute resource management, including single user and multi user problems in single base station environment. Specifically, in Reference [13], the authors mainly consider the computing offloading problem of multi terminal users in single base station environment, which are not aimed at multi base stations environment. In Reference [8], the problem of single user computing offloading is considered, in which MEC resources are not limited. In Reference [15], a MEC system composed of mobile devices and heterogeneous edge servers supporting various wireless access technologies is studied. In Reference [16], the problem of partial offload scheduling and resource allocation in multi task MEC system is studied, and the problem of partial offload scheduling and power allocation in single user MEC system is proposed. In Reference [17,18], although the authors take into account the computing offloading of mobile users in multiple cells,

the main work is focused on the allocation of wireless and computing resources within one base station, without consideration of resource sharing and assistance between the base stations.

The authors of Reference [19] studied the single terminal user scheduling its computing task into a number of smart devices in a number of adjacent environments, and the execution of the task depends on the coordination of multiple devices. The above research work does not focus on the mobile communication network environment, the proposed cooperation concept is limited to the end user centric collaboration, and there is no resource sharing and joint allocation among the intelligent devices.

The architecture design of the future mobile communication system covers the cooperative function between the base stations. Through close collaboration, the base stations in the network can be virtualized as a resource whole and serve each end user flexibly in distributed computing and distributed storage. Therefore, the cooperation between base stations needs to be studied in depth. The technology of resource management improves the utilization ratio of the whole network resources and improves the service quality of users.

2.3. Existing Offloading / Caching Decision Algorithms

In Reference [20], the author considers the multi-user computation offloading problem in MEC system as an evolutionary game model, and proposes an evolutionary game algorithm based on reinforcement learning to achieve efficient computing offloading. However, IoT devices need to adjust their own strategies through continuous evolution and trial and error to maximize the fitness value, which will cause the time complexity of the algorithm is relatively high. Similarly, there are also some problems in the research of literature [21,22] based on game theory. In addition, a large number of mobile edge computing resources managements based on convex optimization (such as References [23,24]), non convex optimization (such as References [25,26]), combinatorial optimization (such as Referencse [27,28]), evolutionary algorithm (such as Reference [29]), etc. Although some of the authors, besides the basic algorithms, also provide approximate and simplified algorithms (such as Reference [24]), these algorithms still need to occupy a certain amount of computing time, especially in large-scale users and large-scale tasks environment, the algorithm cost is still the main problem that affects the real time of decision-making.

The resource management algorithm determines the computing offloading requests and data caching requests in the execution terminal, and the algorithm's own overhead will directly affect the real-time performance of the decision, especially for the time delay sensitive task processing.

2.4. Lack of Resource Management Technology for Cloud

The high cloud server load will seriously reduce the performance and user experience of the mobile edge computing system. Most of the research work on cloud loss mainly focuses on the distributed cloud computing field. Cloud resource optimization mainly considers the load balancing questions between multiple cloud servers and clusters, mainly focusing on the virtualization of multiple clouds and resource sharing technology [30–33].

As for the resource optimization for mobile edge computing with the aim of reducing mobile cloud load, only Reference [34] studies the load reduction method under the high cloud server load, and two load recovery strategies are proposed in the engineering perspective. However, the author only considers the task offloading strategy when the server load is too high or damaged, and does not study the load balance in the multi server computing, storage sharing, and the network transmission equilibrium.

In spite of this, the research on mobile communication network environment is still relatively scarce. In mobile communication network environment, the base station is the edge node of the cloud server. The problems of cloud oriented resource management needs to be solved mainly by: (1) how to use the computing resources of the base station to divert the cloud computing task into the base station; (2) how to use the storage resources of

the base station to divert the cloud data storage to the base station cache, so as to reduce the cloud computing and storage load, reduce the network data transmission load and improve the system performance.

3. System Model

We assume the number of all MTs and MEC-NBS is M and K, respectively. The ith ($i \in \mathcal{M} = \{1, 2, \ldots, M\}$) MT is defined as m_i, and kth ($k \in \mathcal{K} = \{1, 2, \ldots, K\}$) MEC-NBS is defined as n_k. An MT can run multiple mobile applications at the same time, and each application may contain multiple tasks. The number of all types of tasks is assumed to be R, the jth ($j \in \mathcal{R} = \{1, 2, \ldots, R\}$) type of tasks is defined as r_j. Moreover, the data storage of MEC-BS and n_k is defined as d and d_k ($k \in \mathcal{K}$), respectively.

We consider that the request from each MT is a Poisson process, and ϵ_i represents the request rate of m_i, that is, ϵ_i tasks are generated by m_i per second. The request ratio of m_i to the task r_j is $p_{i,j}$ ($p_{i,j} \in [0, 1]$), and the proportion is different for different tasks. Note that $\sum_{j \in \mathcal{R}} p_{i,j} = 1$ because $p_{i,j}$ actually represents the proportion of r_j in the tasks generated by m_i.

We assume that the request message length and the response message length are fixed and roughly equal in the offloading signal transmission or data caching transmission among the MT, MEC-BS, MEC-NBS, and C. Each task r_j is profiled by an ordered vector $< c_j, v_j, w_j, z_j, q_j >$, which is characterized by: (1) c_j, the amount of computation needed to complete r_j; (2) v_j, the size of the offloading request (including necessary description and parameters of r_j) for r_j; (3) w_j, the size of the offloading response (including the result of r_j's execution) for r_j; (4) z_j, the size of the data caching request (including necessary description and parameters of the data required to execute r_j) for r_j; (5) q_j, the size of the data caching response (including the data required to execute r_j) for r_j.

The computing task r_j has 2 ways to completing : execute it at C or offload it to MEC stations. If task r_j is determined to execute at C, it will increase the computing load of the mobile cloud and affect the performance of the cloud server. It will also suffer the offloading-signal transmission delay from the MT to the MEC-BS and the MEC-BS to C. If the MEC-BS chooses to offload r_j to MEC stations, it will greatly alleviate the cloud pressure, and MEC stations have a certain degree of computational power, the efficiency of the computing task can also be guaranteed, but, at the same time, it may suffer from the time consumption caused by data request and response transmission between MEC stations and C. If the MEC-BS chooses to cache the data at MEC stations in advance, it will avoid the time consumption of the offloading-signal and data-signal transmission. However, data caching will take time, and MEC stations have limited storage space.

When r_j is offloaded to MEC stations, it can be executed at MEC-BS, or be further offloaded to a MEC-NBS(the offloading probability of all the n_k is equal). For task r_j, we define the offloading probability from m_i to MEC-BS as $\alpha_{i,j} | (i \in \mathcal{M},\ j \in \mathcal{R})$ and from m_i to n_k as $\beta_{i,j,k} | (i \in \mathcal{M},\ j \in \mathcal{R},\ k \in \mathcal{K})$, so that the offloading probability from m_i to C is $(1 - \alpha_{i,j} - \sum_{k \in \mathcal{K}} \beta_{i,j,k})$. Moreover, we define the caching probability from C to MEC-BS as $\sigma_j | (j \in \mathcal{R})$, from C to MEC-NBS as $\pi_{j,k} | (j \in \mathcal{R},\ k \in \mathcal{K})$.

There are many symbols used by our system model, so we add notions to distinguish them. "O" means "Offloading", "D" means "Data caching", and "2" means "to". For example, "ONBS2BS" expresses the symbol is related to the issue "computation Offloading from a MEC-NBS to the MEC-BS". "DC2NBS" expresses the symbol is related to the issue "Data caching from the Cloud to a MEC-NBS".

3.1. Computation Model

(1) Execution at C

All the tasks which executed at C share the computation resources of it.

By defining the service rate of C as η, if r_j is selected to be executed at C, the time consumed by completing r_j is

$$t_{i,j}^{\mathrm{C}} = \frac{p_{i,j}\epsilon_i c_{i,j}}{\eta - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}\left((1-\alpha_{i,j}-\sum_{k\in\mathcal{K}}\beta_{i,j,k})p_{i,j}\lambda_i c_j\right)}, \tag{1}$$

where the denominator is the stable processing speed [35] (amount of computation processed per second) of C. $\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}\left((1-\alpha_{i,j}-\sum_{k\in\mathcal{K}}\beta_{i,j,k})p_{i,j}\epsilon_i c_j\right)$ is the total amount of computation of C's tasks executed per second which was determined to be offloaded to C. It can be observed that, with the increase of the tasks executed at C, the processing speed of C is decreasing. Note that $\eta - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}\left((1-\alpha_{i,j}-\sum_{k\in\mathcal{K}}\beta_{i,j,k})p_{i,j}\epsilon_i c_j\right) > 0$ is the hard constraint [35] of Formula (3), which means the tasks' arriving rate cannot exceed C's service rate.

(2) Execution at MEC-BS

All the tasks which offloaded to a MEC-BS from C share the computation resources of the MEC-BS.

By defining the service rate of MEC-BS as ϕ, if r_j is selected to be executed at MEC-BS, the time consumed by completing r_j is

$$t_{i,j}^{\mathrm{BS}} = \frac{p_{i,j}\epsilon_i c_{i,j}}{\phi - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\alpha_{i,j}p_{i,j}\epsilon_i c_j)}, \tag{2}$$

where the denominator is the stable processing speed [35] (amount of computation processed per second) of MEC-BS. $\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\alpha_{i,j}p_{i,j}\epsilon_i c_j)$ is the total amount of computation of MEC-BS's tasks executed per second which was determined to be offloaded to MEC-BS. It can be observed that, with the increase of the tasks executed at the MEC-BS, the processing speed of MEC-BS is decreasing. Note that $\phi - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\alpha_{i,j}p_{i,j}\epsilon_i c_j) > 0$ is the hard constraint [35] of Formula (3), which means the tasks' arriving rate cannot exceed MEC-BS's service rate.

(3) Execution at MEC-NBS

All the tasks which offloaded to a MEC-NBS from C share the computation resources of the MEC-NBS.

By defining the service rate of n_k as ϕ_k, if r_j is selected to be executed at n_k, the time consumed by completing r_j is

$$t_{i,j,k}^{\mathrm{NBS}} = \frac{p_{i,j}\epsilon_i c_{i,j}}{\phi_k - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\beta_{i,j,k}p_{i,j}\epsilon_i c_j)}, \tag{3}$$

where the denominator is the stable processing speed [35] (amount of computation processed per second) of n_k. $\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\beta_{i,j,k}p_{i,j}\epsilon_i c_j)$ is the total amount of computation of n_k's tasks executed per second which was determined to be offloaded to MEC-NBS. It can be observed that, with the increase of the tasks executed at the MEC-NBS, the processing speed of n_k is decreasing. Note that $\phi_k - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\beta_{i,j,k}p_{i,j}\epsilon_i c_j) > 0$ is the hard constraint [35] of Formula (3), which means the tasks' arriving rate cannot exceed n_k's service rate.

3.2. Transmission Model

(1) Communications between C and MEC-BS

All the MEC stations, including MEC-BS and MEC-NBS, under a C's coverage share the wireless resources of it. In this paper, the impacts of inter-station and intra-station interferences caused by computation offloading have been ignored because of the extremely tiny sizes of them. There are two types of communications between C and MEC-BS (offloading-signal transmission and data-signal transmission, respectively).

We define the data-signal transmission rate from MEC-BS to C as s_i^{BS2C}. Then, the time consumed by sending the offloading request of r_j from MEC-BS to C can be defined as follows, if r_j is selected to be offloaded.

$$t_{i,j}^{\mathrm{OBS2C}} = \frac{p_{i,j}\epsilon_i v_j}{s_i^{\mathrm{BS2C}}}. \tag{4}$$

The data transmission rate from C to MEC-BS is denoted by s_i^{C2BS}. Then, we have the time consumed by receiving the offloading response of r_j from C to MEC-BS:

$$t_{i,j}^{\mathrm{OC2BS}} = \frac{p_{i,j}\epsilon_i w_j}{s_i^{\mathrm{C2BS}}}. \tag{5}$$

As for data caching transmission, MEC-BS uses the Cooperative Resource Management Algorithm to determine the most profitable data caching which is required by task r_j, the time consumed by sending the data caching request of r_j from MEC-BS to C can be defined as follows:

$$t_{i,j}^{\mathrm{DBS2C}} = \frac{p_{i,j}\epsilon_i z_j}{s_i^{\mathrm{BS2C}}}. \tag{6}$$

Similarly,we have the time consumed by receiving the data caching response of r_j from C to MEC-BS:

$$t_{i,j}^{\mathrm{DC2BS}} = \frac{p_{i,j}\epsilon_i q_j}{s_i^{\mathrm{C2BS}}}. \tag{7}$$

(2) Communications between C and MEC-NBS

MEC-NBS can also use the Cooperative Resource Management Algorithm to determine the most profitable data caching which is required by task r_j. We define the data-signal transmission rate from n_k to C as s_k^{NBS2C} the time consumed by sending the data caching request of r_j from n_k to C can be defined as follows:

$$t_{i,j,k}^{\mathrm{DNBS2C}} = \frac{p_{i,j}\epsilon_i z_j}{s_k^{\mathrm{NBS2C}}}. \tag{8}$$

The data transmission rate from C to n_k is denoted by s_k^{C2NBS}. Then, we have the time consumed by receiving the data caching response of r_j from C to n_k:

$$t_{i,j,k}^{\mathrm{DC2NBS}} = \frac{p_{i,j}\epsilon_i q_j}{s_k^{\mathrm{C2NBS}}}. \tag{9}$$

(3) Communications between MEC-BS and MEC-NBS

For the sake of data security, we assume that data-signal cannot be transmitted between the MEC stations, including MEC-BS and MEC-NBS. Accordingly, there is only one type of communications between MEC-BS and n_k, which is the offloading-signal transmission. We define the offloading-signal transmission rate from MEC-BS to n_k through the wired connection between them as s_k^{BS2NBS}. Reversely, s_k^{NBS2BS} represents the offloading-signal transmission rate from n_k to MEC-BS through the connection.

The time consumed by transmitting the offloading request of r_j from MEC-BS to n_k is expressed as

$$t_{i,j,k}^{\mathrm{OBS2NBS}} = \frac{p_{i,j}\epsilon_i v_j}{s_k^{\mathrm{BS2NBS}}}. \tag{10}$$

Similarly, the time consumed by transmitting the offloading response of r_j from n_k to MEC-BS is expressed as

$$t_{i,j,k}^{\mathrm{ONBS2BS}} = \frac{p_{i,j}\epsilon_i w_j}{s_k^{\mathrm{NBS2BS}}}. \tag{11}$$

(4) Communications between MEC-BS and MTs

All the MTs under a MEC-BS's coverage share the wireless resources of it. In this paper, we ignore the computing power of every MTs, so that there is only one type of communications between MTs and MEC-BS which is the offloading-signal data transmission.

We define the uplink data transmission rate from m_i to MEC-BS as s_i^{MT2BS}. Then, the time consumed by sending the offloading request of r_j from m_i to MEC-BS can be defined as follows, if r_j is selected to be offloaded.

$$t_{i,j}^{\text{OMT2BS}} = \frac{p_{i,j}\epsilon_i v_j}{s_i^{\text{MT2BS}}}. \tag{12}$$

The downlink data transmission rate from MEC-BS to m_i is denoted by s_i^{BS2MT}. Then, we have the time consumed by receiving the offloading response of r_j from MEC-BS to m_i:

$$t_{i,j}^{\text{OBS2MT}} = \frac{p_{i,j}\epsilon_i w_j}{s_i^{\text{BS2MT}}}. \tag{13}$$

3.3. Optimization Model

The total time consumption for completing r_j includes: (1) the time consumed by executing in the mobile cloud, if r_j is selected to be executed at C; (2) the time consumed by computation offloading, if r_j is selected to be offloaded to MEC-BS; (3) the time consumed by computation offloading, if r_j is selected to be further offloaded to $n_k, \forall k \in \mathcal{K}$.

In (1), the time consumption is generated by transmitting the offloading request of r_j from m_i to MEC-BS, transmitting the offloading request of r_j from MEC-BS to C, executing r_j at C, transmitting the offloading response of r_j from C to MEC-BS, and transmitting the offloading response of r_j from MEC-BS to m_i, that is, $(1 - \sum_{k\in\mathcal{K}} \beta_{i,j,k} - \alpha_{i,j})(t_{i,j}^{\text{OMT2BS}} + t_{i,j}^{\text{OBS2C}} + t_{i,j}^{\text{C}} + t_{i,j}^{\text{OC2BS}} + t_{i,j}^{\text{OBS2MT}})$.

In (2), the time consumption needs to be divided into two situations: (1) MEC-BS has the data caching required by task r_j; (2) MEC-BS does not have the data caching required by task r_j, and it needs to send the data caching request to C to get the data. The two cases are discussed separately as follows:

(1) The time consumption is generated by transmitting the offloading request of r_j from m_i to MEC-BS, executing r_j at MEC-BS, and transmitting the offloading response of r_j from MEC-BS to m_i, that is, $\alpha_{i,j}\sigma_j(t_{i,j}^{\text{OMT2BS}} + t_{i,j}^{\text{BS}} + t_{i,j}^{\text{OBS2MT}})$.

(2) The time consumption is generated by transmitting the offloading request of r_j from m_i to MEC-BS, transmitting the data caching request of r_j from MEC-BS to C, transmitting the data caching response of r_j from C to MEC-BS, executing r_j at MEC-BS, and transmitting the offloading response of r_j from MEC-BS to m_i, that is, $\alpha_{i,j}(1 - \sigma_j)(t_{i,j}^{\text{OMT2BS}} + t_{i,j}^{\text{DBS2C}} + t_{i,j}^{\text{DC2BS}} + t_{i,j}^{\text{BS}} + t_{i,j}^{\text{OBS2MT}})$.

In (3), the time consumption needs to be divided into two situations: (1) n_k has the data caching required by task r_j; (2) n_k does not have the data caching required by task r_j, and it needs to send the data caching request to C to get the data. The two cases are discussed separately as follows:

(1) The time consumption is generated by transmitting the offloading request of r_j from m_i to MEC-BS, transmitting the offloading request of r_j from MEC-BS to n_k, executing r_j at n_k, transmitting the offloading response of r_j from n_k to MEC-BS, and transmitting the offloading response of r_j from MEC-BS to m_i, that is, $\sum_{k\in\mathcal{K}} \beta_{i,j,k}\pi_{j,k}(t_{i,j}^{\text{OMT2BS}} + t_{i,j,k}^{\text{OBS2NBS}} + t_{i,j,k}^{\text{NBS}} + t_{i,j,k}^{\text{ONBS2BS}} + t_{i,j}^{\text{OBS2MT}})$.

(2) The time consumption is generated by transmitting the offloading request of r_j from m_i to MEC-BS, transmitting the offloading request of r_j from MEC-BS to n_k, transmitting the data caching request of r_j from n_k to C, transmitting the data caching response of r_j from C to n_k, executing $h_{i,j}$ at bk, transmitting the offloading response of $h_{i,j}$ from bk to bs, and transmitting the offloading response of $h_{i,j}$ from MEC-BS to m_i, that is, $\sum_{k\in\mathcal{K}} \beta_{i,j,k}(1 - \pi_{j,k})(t_{i,j}^{\text{OMT2BS}} + t_{i,j,k}^{\text{OBS2NBS}} + t_{i,j,k}^{\text{DNBS2C}} + t_{i,j,k}^{\text{DC2NBS}} + t_{i,j,k}^{\text{NBS}} + t_{i,j,k}^{\text{ONBS2BS}} + t_{i,j}^{\text{OBS2MT}})$.

In summary, we have the total time consumption for completing r_j:

$$\begin{aligned} t_{i,j} =&(1-\alpha_{i,j}-\sum_{k\in\mathcal{K}}\beta_{i,j,k})t_{i,j}^{\mathrm{C}}+\alpha_{i,j}t_{i,j}^{\mathrm{BS}}+\sum_{k\in\mathcal{K}}\beta_{i,j,k}t_{i,j,k}^{\mathrm{NBS}} \\ &+\alpha_{i,j}(1-\sigma_j)(t_{i,j}^{\mathrm{DBS2C}}+t_{i,j}^{\mathrm{DC2BS}})+(1-\alpha_{i,j} \\ &-\sum_{k\in\mathcal{K}}\beta_{i,j,k})(t_{i,j}^{\mathrm{OBS2C}}+t_{i,j}^{\mathrm{OC2BS}})+\sum_{k\in\mathcal{K}}\beta_{i,j,k}(\\ &t_{i,j,k}^{\mathrm{OBS2NBS}}+t_{i,j,k}^{\mathrm{ONBS2BS}})+\sum_{k\in\mathcal{K}}\beta_{i,j,k}(1-\pi_{j,k})(\\ &t_{i,j,k}^{\mathrm{DNBS2C}}+t_{i,j,k}^{\mathrm{DC2NBS}})+t_{i,j}^{\mathrm{OMT2BS}}+t_{i,j}^{\mathrm{OBS2MT}} \end{aligned} . \tag{14}$$

Therefore, the total time consumption of all MTs covered by MEC-BS can be formulated as t:

$$t=\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}t_{i,j}. \tag{15}$$

4. Cooperative Resource Management Algorithm

4.1. Optimization Problem

The aim of our algorithm is to minimize the total time consumption t gained by all MTs in $\mathcal{M}$, while ensuring all constraints are not violated. Thus, the corresponding optimization problem can be formulated as

$$\underset{\boldsymbol{\alpha},\beta,\sigma,\pi}{\text{minimize}}\ t, \tag{16}$$

$$\text{subject to } \alpha_{i,j}\in[0,1],\quad \forall i\in\mathcal{M},\forall j\in\mathcal{R}, \tag{17}$$

$$\beta_{i,j,k}\in[0,1],\quad \forall i\in\mathcal{M},\forall j\in\mathcal{R},\forall k\in\mathcal{K}, \tag{18}$$

$$\alpha_{i,j}+\sum_{k\in\mathcal{K}}\beta_{i,j,k}\in[0,1],\quad \forall i\in\mathcal{M},\forall j\in\mathcal{R}, \tag{19}$$

$$\sigma_j\in[0,1],\quad \forall j\in\mathcal{R}, \tag{20}$$

$$\pi_{j,k}\in[0,1],\quad \forall j\in\mathcal{R},\forall k\in\mathcal{K}, \tag{21}$$

$$\eta-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}\left((1-\alpha_{i,j}-\sum_{k\in\mathcal{K}}\beta_{i,j,k})p_{i,j}\epsilon_i c_j\right)>0, \tag{22}$$

$$\mu-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\alpha_{i,j}p_{i,j}\epsilon_i c_j)>0, \tag{23}$$

$$\mu_k-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(\beta_{i,j,k}p_{i,j}\epsilon_i c_j)>0,\quad \forall k\in\mathcal{K}, \tag{24}$$

$$\sum_{j\in\mathcal{R}}(c_j\sigma_j)\leqslant s, \tag{25}$$

$$\sum_{j\in\mathcal{R}}(c_j\pi_{j,k})\leqslant s_k,\quad \forall k\in\mathcal{K}, \tag{26}$$

where constraint (17) is the value range of each $\alpha_{i,j}$, constraint (18) is the value range of each $\beta_{i,j,k}$, constraint (20) is the value range of each σ_j, and constraint (21) is the value range of each $\pi_{j,k}$. As aforementioned, Constraint (19) is the value range of the total probability that r_j is offloaded from C. Constraint (22)–(24) is hard constraint of the offloading queuing systems of C, MEC-BS, and n_k, respectively. Constraint (25) and (26) is hard constraint of the caching queuing systems of MEC-BS and n_k, respectively.

We define the total time consumption without computation offloading and data caching as $\tilde{t}$, and we expand $\tilde{t}$ as

$$\begin{aligned}\tilde{t} &= \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}} t_{i,j}^{C}|\alpha_{i,j}=0,\beta_{i,j,k}=0,\sigma_j=0,\pi_{j,k}=0 \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}\left(\frac{p_{i,j}\epsilon_i c_{i,j}}{\eta-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(p_{i,j}\epsilon_i c_j)}\right)\end{aligned} . \tag{27}$$

We can come to this conclusion that the sufficient condition of $\tilde{t}$ is that $\eta-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(p_{i,j}\epsilon_i c_j)>0$ must hold. By comparing the condition with constraint (22), we have

$$\begin{aligned}\eta-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}\left((1-\alpha_{i,j}-\sum_{k\in\mathcal{K}}\beta_{i,j,k})p_{i,j}\epsilon_i c_j\right)\geqslant \\ \eta-\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{R}}(p_{i,j}\epsilon_i c_j)>0.\end{aligned} \tag{28}$$

Thus, constraint (22) always holds.

4.2. Optimization Algorithm Using Genetic algorithm

To determine the feasible domain of the problem, we transform all the independent variables into one-dimensional column vectors and combine them into a whole one-dimensional column vector. Thus, $\alpha_{i,j}$ is converted to a vector $\boldsymbol{\alpha}_i$ $(i\in\{1,2,\ldots,\mathbf{M}*\mathbf{R}\})$. Similarly, $\beta_{i,j,k}$ = $\boldsymbol{\beta}_i(i\in\{1,2,\ldots,\mathbf{M}*\mathbf{R}*\mathbf{K}\})$, σ_j = $\boldsymbol{\sigma}_i(i\in\{1,2,\ldots,\mathbf{R}\})$, $\pi_{j,k}$ = $\boldsymbol{\pi}_i(i\in\{1,2,\ldots,\mathbf{R}*\mathbf{K}\})$. So, we can combine them into $\mathbf{H_i}$ $(\mathbf{i}\in\{\mathbf{1},\mathbf{2},\ldots,\mathbf{M}*\mathbf{R}+\mathbf{M}*\mathbf{R}*\mathbf{K}+\mathbf{R}+\mathbf{R}*\mathbf{K}\})$ in order of $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\sigma},\boldsymbol{\pi}$, where $\mathbf{H}$ is the feasible region of the problem.

In combination with (16) and constraints above, we derive the convexity and concavity of (16). By splitting (16) and judging the positive definiteness of each part of Hessian matrix, we obtain that (16) is a non-convex and non-concave function due to the existence of several saddle points. Therefore, if we use the traditional numerical optimization algorithm (such as interior point method) to get the optimal solution of (16), it is easy to fall into local minimum. The "dead cycle" phenomenon occurs because of the minimal part trap, which makes the iteration impossible, and only the local optimal solution is obtained instead of global optimal solution.

Genetic algorithm overcomes this shortcoming very well, which is a global optimization algorithm. Due to the evolutionary characteristics of genetic algorithm, the intrinsic properties of the optimization problem in the process of searching element have little effect on the final optimization result. Moreover, the ergodicity of the evolutionary operator of genetic algorithm makes it very effective to search element with probabilistic global significance, which matches the optimization object and optimization goal of our paper very well.

In addition, compared with the accurate algorithm, the approximate algorithm has less time consumption and space consumption, so that it can guarantee the overall system performance. In this paper, we choose the genetic algorithm to solve the optimization problem.

We use $\mathbf{H}=0$ as the initial generation, and $\mathbf{H}$ should strictly adhere to the above constraints.

Now, we can give the optimization algorithm using Genetic algorithm, which is shown in Algorithm 1.

Algorithm 1 Algorithm Solving Optimization Problem

Parameters - Population Size = 150
- GA Generations = 100
- Crossover ratio = 1.2

Selection Function: Chooses parents by simulating a roulette wheel, in which the area of the section of the wheel corresponding to an individual is proportional to the individual's expectation. The algorithm uses a random number to select one of the sections with a probability equal to its area.

Crossover Function: Returns a child that lies on the line containing the two parents, a small distance away from the parent with the better fitness value in the direction away from the parent with the worse fitness value. You can specify how far the child is from the better parent by the parameter Ratio, which appears when you select Heuristic. The default value of Ratio is 1.2.

Mutate Function: Randomly generate directions that are adaptive with respect to the last successful or unsuccessful generation. The mutation chooses a direction and step length that satisfies bounds and linear.

Begin
T = 0;
Initialize **H**(t) ;
Evaluate **H**(t);
While not finished do
Begin
T = t + 1;
Select **H**(t) from **H**(t − 1);
Crossover in **H**(t);
Mutate in **H**(t);
Evaluate **H**(t);
End
End

4.3. Cooperative Resource Management Algorithm

The data cached from C and the computation offloading for each MT is managed by MEC-BS using Cooperative Resource Management Algorithm, which is responsible for monitoring and collecting the information (computation offloading requests and responses, data caching requests and responses, parameters of MTs, MEC-NBS, and C), running the optimization algorithm, and sending the optimization result to C and each MT.

MEC-BS will send the selection probability in the optimization result to the MT and C to determine the place (C, MEC-BS, or MEC-NBS), which is selected to execute the task or cache the data. At initialization of the system, all MTs in $\mathcal{M}$ upload their required parameters, including $\forall i \in \mathcal{M}, \forall j \in \mathcal{R}$, $p_{i,j}$, s_i^{MT2BS}, s_i^{BS2MT}, ϵ_i, and the information of all its tasks ($< c_j, v_j, w_j, z_j, q_j >$), to MEC-BS. MEC-BS also collects the required parameters of MEC-NBS, including $\forall k \in \mathcal{K}, \phi_k$, s_k^{BS2NBS}, and s_k^{NBS2BS}. The required paramenters of C need to be sent to MEC-BS, as well, including eta, s_i^{BS2C}, and s_i^{C2BS}, and parameters between C and MEC-NBS s_k^{NBS2C} and s_k^{C2NBS} should also be sent.

During the system running, as for MTs, MEC-NBS, and C, once the value of any parameters of them has changed, they need to report the new value to MEC-BS. So, it can ensure the real-time and correctness of the parameter values in MEC-BS.

MEC-BS monitors periodic observation of changes in all parameters. It will run Algorithm 1 in the next period, if any parameter has changed. Periodic monitoring reduces the execution frequency of the algorithm, while ensuring the timeliness of the optimization results.

The flowchart of the Cooperative Resource Management Algorithm is shown in Figure 3. The details of the algorithm are described in Algorithm 2 for MT side, Algorithm 3 for C side, Algorithm 4 for MEC-NBS side, and Algorithm 5 for MEC-BS side. The four

algorithms are composed of twelve loops, *L1-L12*; these loops are deployed into separate processes and executed in parallel during the running period of the system.

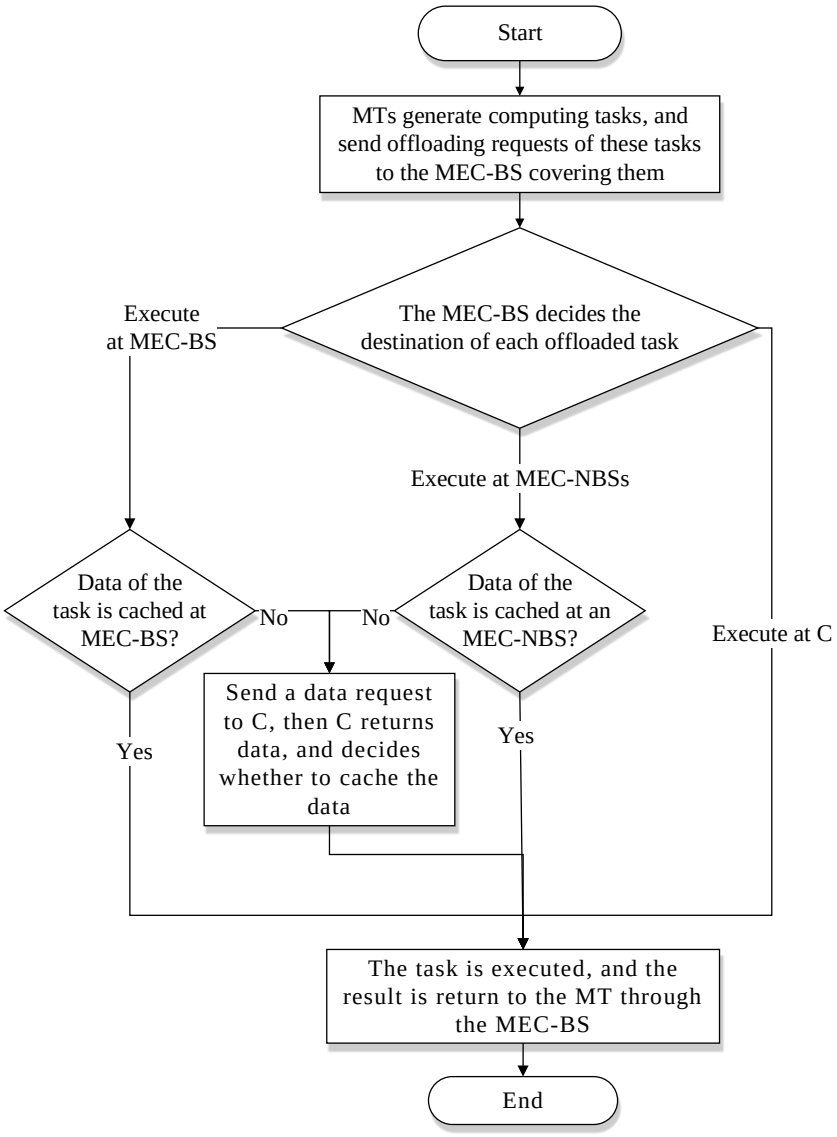

Figure 3. Flowchart of the Cooperative Resource Management Algorithm.

Algorithm 2 Cooperative Resource Management Algorithm at m_i side

Initial: set $\mathbf{H} = 0$, send $p_{i,j}$, ϵ_i, s_i^{MT2BS}, s_i^{BS2MT} and the profiles of all its tasks to MEC-BS.
L1:
if any required parameters of m_i changes **then**
m_i sends the new value of the parameter to MEC-BS;
end if

L2:
if m_i receives $\mathbf{H_i}$ from MEC-BS **then**
m_i updates the value of $\mathbf{H_i}$ stored in $\mathbf{H}$;
end if

L3:
if m_i has a new r_j **then**
m_i generates a random number $h \in [0, 1]$ based on Uniform distribution;
$b1 = 0, b2 = 0$;
for $b = 1$ to $(M * R + M * R * K)$ **do**
if $h \leq \mathbf{H_n}$ **then**
if $b \leq (M * R)$ **then** $b1 = b$;
else $b2 = b$;
end if
break;
end if
end for
if $(b1 == 0)\&(b2 == 0)$ **then**
m_i executes r_j at C;
else if $(b1 != 0)$ **then**
m_i offloads r_j to MEC-BS;
else
m_i offloads r_j to n_{b2};
end if

L4:
if m_i has a new r_j **then**
m_i generates a random number $c \in [0, 1]$ based on Uniform distribution;
$b3 = 0, b4 = 0$;
for b=$(M * R + M * R * K)$ to $(M * R + M * R * K + R + R * K)$ **do**
if $c \leq \mathbf{H_b}$ **then**
if $b \leq (M * R + M * R * K + R)$ **then** $b3 = b$;
else $b4 = b$;
end if
break;
end if
end for
if $(b3 == 0)\&(b4 == 0)$ **then**
the data required by r_j does not be cached anywhere;
else if $(b3 != 0)$ **then**
the data required by r_j is cached in MEC-BS from C;
else
the data required by r_j is cached in n_{b4} from C;
end if
end if

Algorithm 3 Cooperative Resource Management Algorithm at C side

Initial: send η, s_i^{BS2C}, s_i^{C2BS}, s_k^{NBS2C} and s_k^{C2NBS} to MEC-BS.
L5:
if any required parameters of C changes **then**
C sends the new value of the parameter to MEC-BS;
end if

L6:
if r_j is decided to be offloaded to C **then**
C executes r_j;
C returns the offloading response of r_j to MEC-BS;
end if

L7:
if the data required by r_j should be cached in MEC-BS from C **then**
MEC-BS sends the data caching request of r_j to C;
C returns the data caching response of r_j to MEC-BS;
end if

L8:
if the data required by r_j should be cached in n_k from C **then**
n_k sends the data caching request of r_j to C;
C returns the data caching response of r_j to n_k;
end if

Algorithm 4 Cooperative Resource Management Algorithm at n_k side ($k \in \mathcal{K}$)

Initial: send ϕ_k, s_k^{NBS2BS} and s_k^{BS2NBS} to MEC-BS.
L9:
if any required parameters of n_k changes **then**
n_k sends the new value of the parameter to MEC-BS;
end if

L10:
if r_j is decided to be offload to n_k and n_k cached the data of r_j in advance **then**
n_k executes r_j;
n_k returns the offloading response of r_j to MEC-BS;
else if r_j is decided to be offload to n_k and n_k did not cache the data of r_j in advance **then**
n_k sends the data request of r_j to C;
C returns the data response of r_j to n_k;
n_k executes r_j;
n_k sends the offloading response of r_j to MEC-BS;
end if
end if

Algorithm 5 Cooperative Resource Management Algorithm at MEC-BS side

Initial: collect required parameters from MTs, MEC-NBS, C, and itself, then store their values. *L11* (runs periodically): **if** MEC-BS receives a new value of the required parameters **then**
 MEC-BS runs Algorithm 1 and get optimal result **H**;
 for each $\mathbf{H_i}$ in **H do**
  if the value of $\mathbf{H_i}$ is updated **then**
   MEC-BS sends the new value of $\mathbf{H_i}$ to m_i;
  end if
 end for
end if

L12:
if MEC-BS receives the offloading request of r_j from m_i **then**
 if r_j is decided to be offload to MEC-BS and MEC-BS cached the data of r_j in advance **then**
  MEC-BS executes r_j;
  MEC-BS sends the offloading response of r_j to m_i;
 else if r_j is decided to be offload to MEC-BS and MEC-BS did not cache the data of r_j in advance **then**
  MEC-BS sends the data request of r_j to C;
  C returns the data response of r_j to MEC-BS;
  MEC-BS executes r_j;
  MEC-BS sends the offloading response of r_j to m_i;
 end if
end if

L13:
if MEC-BS receives the offloading response of r_j sent from C **then**
  MEC-BS forwards the offloading response to m_i;
end if

L14:
if MEC-BS receives the offloading response of r_j sent from n_k ($\forall k \in \mathcal{K}$) **then**
  MEC-BS forwards the offloading response to m_i;
end if

(*L1*): if any one of $p_{i,j}$, ϵ_i, s_i^{MT2BS}, s_i^{BS2MT} and the profiles of all its tasks to MEC-BS changes, m_i will send the new value of the parameter to MEC-BS;

(*L2*): if m_i receives $\mathbf{H_i}$ from MEC-BS, it will update the value of $\mathbf{H_i}$ stored in **H**;

(*L3*): when m_i has a new r_j, as for computing offloading, firstly, it produces a random number $h \in [0, 1]$ based on Uniform distribution. Then, it compares h with the offloading probabilities in H. If h falls into the interval representing $b1$, then m_i will offload r_j to MEC-BS by sending offloading request and receiving offloading response. Similarly, if h falls into the interval representing $b2$, then m_i will offload r_j to n_k. Else, r_j will be offloaded to C.

(*L4*): when m_i has a new r_j, as for data caching, firstly, it produces a random number $c \in [0, 1]$ based on Uniform distribution. Then, it compares c with the data caching probabilities in H. If c falls into the interval representing $b3$, then m_i will cache the data requested by r_j in MEC-BS by sending data caching request and receiving data caching response. Similarly, if c falls into the interval representing $b4$, then m_i will cache the data requested by r_j in n_k. Else, the data requested by r_j will not be cached.

(*L5*): if any one of η, s_i^{BS2C}, s_i^{C2BS}, s_k^{NBS2C} and s_k^{C2NBS} changes, C will send the new value of the parameter to MEC-BS;

(*L6*): if C receives the offloading request of r_j from MEC-BS, C will execute r_j, then return the offloading response to MEC-BS;

(*L7*): if C receives the data caching request of r_j to MEC-BS, C will return the data caching response of r_j to MEC-BS;

(*L8*): if C receives the data caching request of r_j to n_k, C will return the data caching response of r_j to n_k;

(*L9*): if any one of ϕ_k, s_k^{NBS2BS} and s_k^{BS2NBS} changes, n_k will send the new value of the parameter to MEC-BS;

(*L10*): if n_k receives the offloading request of r_j from MEC-BS and n_k cached the data of r_j in advance, n_k will execute r_j, then return the offloading response to MEC-BS. If n_k receives the offloading request and n_k did not cache the data of r_j in advance, n_k will send the data request of r_j to C, then C will return the data response of r_j to n_k. After that n_k will execute r_j, then return the offloading response to MEC-BS.

(*L11*): L11 runs periodically. If MEC-BS receives any new values of the required parameters from MTs, MEC-NBS, C, and itself, in last period, MEC-BS will run Algorithm 1 and obtains the optimal solution **H**. Then, MEC-BS checks each element in **H**. MEC-BS will send the new value to the corresponding MT, if **H** is updated.

(*L12*): if MEC-BS receives the offloading request of r_j offloaded from m_i and MEC-BS cached the data of r_j before, MEC-BS will execute r_j, then return the offloading response to m_i. If MEC-BS receives the offloading request and it did not cache the data of r_j before, MEC-BS will send the data request of r_j to C, then C will return the data response of r_j to MEC-BS. After that MEC-BS will execute r_j, then return the offloading response to m_i.

(*L13*): if MEC-BS receives the offloading response of r_j sent from C, MEC-BS will forward the offloading response to m_i.

(*L14*): if MEC-BS receives the offloading response of r_j sent from n_k ($\forall k \in \mathcal{K}$), MEC-BS will forward the offloading response to m_i.

5. Simulations and Performance Evaluations

We carried out extensive simulation of the system, and adopted the strategy of averaging multiple sets of random data to eliminate the measurement errors in the simulation process. We have adopted two contrasting strategies, one of which is the strategy without data cache, and the other is the strategy of neither data cache nor computation offloading. The three strategies are compared to analyze the effect of each strategy on the response time delay of the system from (1) M, the number of MTs; (2) R, the number of a m_i's r_j; (3) K, the number of MEC-NBS; (4) ϕ_k, the service rate of $n_k, k \in \mathcal{K}$; (5) $s_k^{\text{C2NBS}}, \forall k \in \mathcal{K}$, data transmission rate between C and MEC-NBS; (6) $s_k^{\text{NBS2C}}, \forall k \in \mathcal{K}$, data transmission rate between MEC-NBS and C; (7) s_k^{BS2NBS} and $s_k^{\text{NBS2BS}}, \forall k \in \mathcal{K}$, data transmission rate between MEC-BS and MEC-NBS. The result in each of the 8 scenarios is obtained from 50 repeated simulations using different random seeds. The simulation results verify the effectiveness of our strategy. It can well meet the requirements of minimizing the service response delay and improving the service quality of the resource scheduling.

Table 1 lists the settings of the parameters used in simulations. Parameter values are configured based on the table unless stated clearly.

Table 1. The parameters used in simulations and their default values.

Name	Value	Name	Value
M	[30, 100]	R	[1, 8]
K	[2, 9]	p_{ij}	[0, 1]
d	2.4×10^7 MB	d_k	$[2.2, 2.4] \times 10^7$ MB
ϵ_i	[0.1, 1] /s	$c_{i,j}$	$[5, 9] \times 10^4$ MB
v_j	[10, 20] MB	w_j	[10, 20] MB
z_j	[30, 55] MB	q_j	[150, 360] MB

Table 1. *Cont.*

Name	Value	Name	Value
$\phi_k, k \in \mathcal{K}$	$[2.4, 2.7] \times 10^6$ MBPS	ϕ	2.7×10^6 MBPS
η	3.5×10^6 MBPS	s_i^{BS2C}	[20, 30] MB/S
s_i^{C2BS}	[60, 80] MB/s	s_k^{NBS2C}	[20, 30] MB/s
s_k^{C2NBS}	[60, 80] MB/s	s_k^{BS2NBS}	[20, 30] MB/s
s_k^{NBS2BS}	[20, 30] MB/s	s_i^{MT2BS}	[10, 15] MB/s
s_i^{BS2MT}	[10, 15] MB/s		

Table 2 lists the three schemes in MATLAB simulation. We use genetic algorithm to get the minimum system delay of OMEC, and simulate the three strategies mentioned above. The results of simulating are shown in the figures.

Table 2. The three schemes in MATLAB simulation.

Name	Value
OCMEC	Our offloading and caching strategy in MEC environment
OMEC	Only offloading strategy in MEC environment
NMEC	No MEC environment

5.1. Different Numbers of MTs

As shown in Figure 4, we measured the total time consumption by the system in completing all tasks under the 3 strategies with different numbers of MTs. In simulations, we increase M from 30 to 100, while other settings are listed in Table 1, except fixed values $R = 6$, $K = 5$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{NBS2C}} = 30$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$.

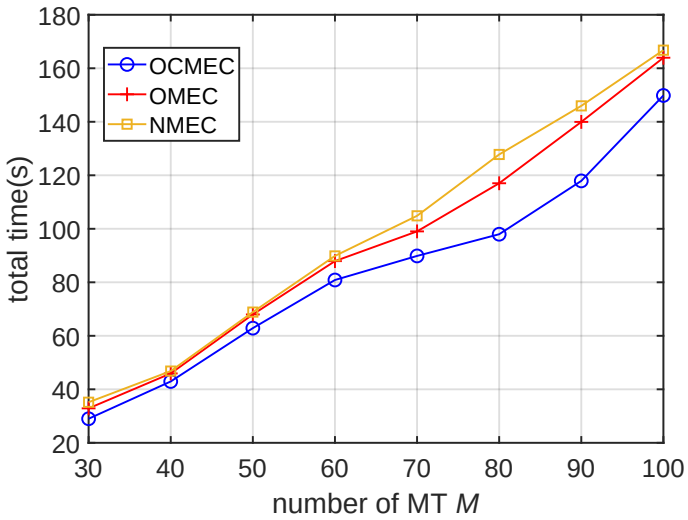

Figure 4. f gained by the 3 schemes with the increase of M.

When M increases, the total time for the system to complete the tasks increases correspondingly, these 3 strategies are in accordance with the law. At the same time, the total time consumed by OMEC and OCMEC is always lower than NMEC, OCMEC is always lower than OMEC. So, computing offloading and data caching can reduce the system delay.

With the increase of M, the total load of the system increases. When $M \in \{30, 40, 50\}$, because of the low task load, the gain effect of OCMEC on reducing the delay is relatively

low; when $M \in \{50, 60, 70, 80\}$, the task load increases continuously, and the total time-consuming gap between OCMEC and NMEC is increasing, and the gain effect of reducing the delay is getting better and better. When $M \in \{80, 90, 100\}$, the total load exceeds the capacity of MEC-BS and MEC-NBS, and they cannot fully meet the calculation requirements of the tasks, so the total time gap gradually narrows, and the gain effect for reducing the delay gradually decreases.

5.2. Different Numbers of MTs' Tasks

As shown in Figure 5, we measured the total time consumption by the system in completing all tasks under the 3 strategies with different numbers of MTs' Tasks. In simulations, we increase R from 1 to 8, while other settings are listed in Table 1, except fixed values $M = 60$, $K = 7$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{NBS2C}} = 30$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$. When R increases, the total time for the system to complete the tasks increases correspondingly, these 3 strategies are in accordance with the law. At the same time, the total time consumed by OMEC and OCMEC is always lower than NMEC, and OCMEC is always lower than OMEC. So, computing offloading and data caching can reduce the system delay.

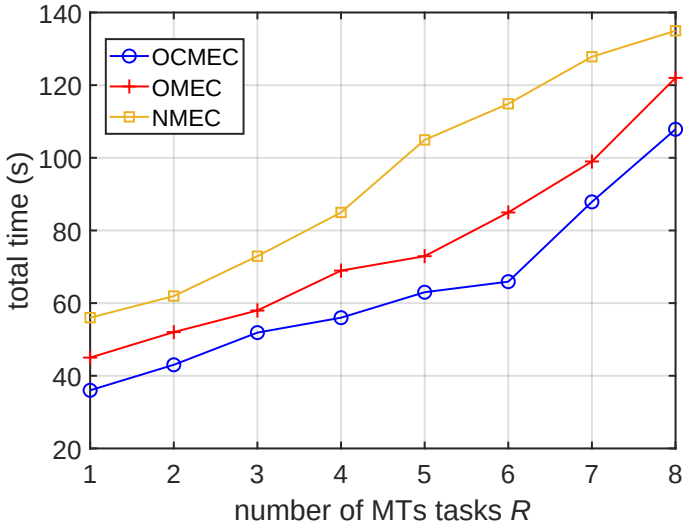

Figure 5. f gained by the 3 schemes with the increase of R.

With the increase of R, the total load of the system increases. When $R \in \{1, 2, 3\}$, because of the low task load, the gain effect of OCMEC on reducing the delay is relatively low; when $R \in \{3, 4, 5, 6\}$, the task load increases continuously, and the total time-consuming gap between OCMEC and NMEC is increasing, and the gain effect of reducing the delay is getting better and better. When $R \in \{6, 7, 8\}$, the total load exceeds the capacity of MEC-BS and MEC-NBS, and they cannot fully meet the calculation requirements of the tasks, so the total time gap gradually narrows, and the gain effect for reducing the delay gradually decreases.

Therefore, the strategy proposed in this paper can perform better when the number of users is larger, and the proposed base stations data caching and computational offloading strategies can effectively reduce the end-user service response latency.

5.3. Different Numbers of MEC-NBS

As shown in Figure 6, we measured the total time consumption by the system in completing all tasks under the 3 strategies with different numbers of MEC-NBS. In simulations, we increase K from 2 to 9, while other settings are listed in Table 1, except fixed values $M = 60$, $R = 6$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{NBS2C}} = 30$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$.

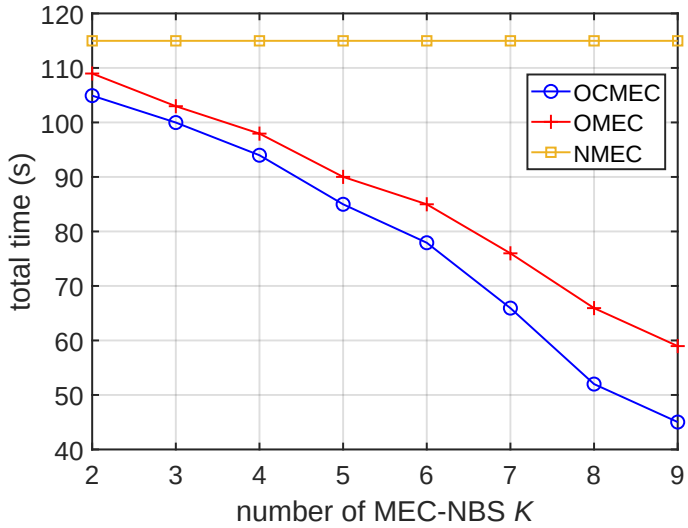

Figure 6. f gained by the 3 schemes with the increase of K.

When K is increased, the total time to complete tasks is decreased correspondingly under OMEC and OCMEC strategies. The curve of NMEC is flat since the further offloading and data caching are disabled, so K has no effect on NMEC. At the same time, the total time consumed by OMEC and OCMEC is always lower than NMEC, and OCMEC is always lower than OMEC. So, computing offloading and data caching can reduce the system delay.

With the increase of K, the system can accept more tasks and provide better performance through computing offloading and data caching, the total time-consuming gap between OCMEC and NMEC is increasing, and the gain effect of reducing the delay is getting better and better. But we still observe that, when the number of K increases to a certain threshold, the optimization effect for time consumption slows down gradually.

5.4. Different Service Rates of MEC-NBS

As shown in Figure 7, we measured the total time consumption by the system in completing all tasks under the 3 schemes with different service rates of MEC-NBS. In simulations, we increase ϕ_k from 2.2×10^6 to 2.7×10^6 MIPS, while other settings are listed in Table 1, except fixed values $M = 60$, $R = 6$, $K = 6$, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{NBS2C}} = 30$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$.

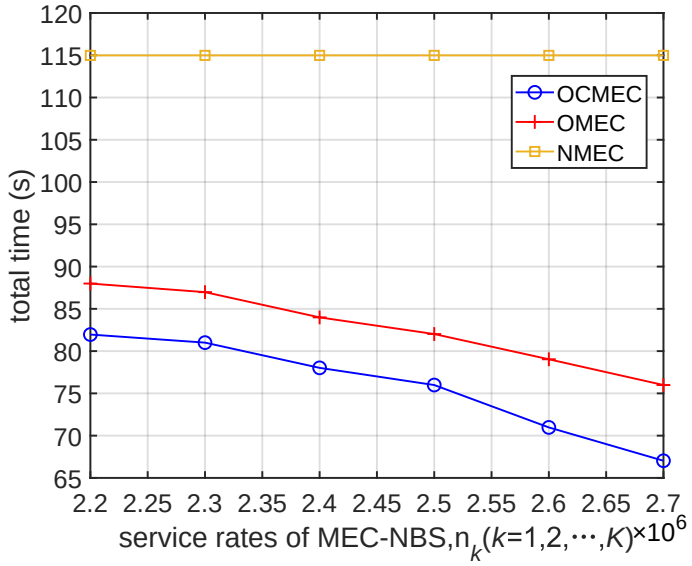

Figure 7. f gained by the 3 schemes with the increase of ϕ_k, the service rate of n_k, $k = 1, 2, \ldots, K$.

ϕ_k represents the computation capability of MEC-NBS. Similar to Figure 6, when ϕ_k is increased, total utilization time of the system is decreased under OMEC and OCMEC strategies. The line of NMEC is straight since ϕ_k has no effect on it. The total time consumed by OMEC and OCMEC is always lower than NMEC, OCMEC is always lower than OMEC. So, computing offloading and data caching can reduce the system delay.

With the increase of ϕ_k, the system can accept more tasks and provide better performance through computing offloading and data caching, the total time-consuming gap between OCMEC and NMEC is increasing, and the gain effect of reducing the delay is getting better and better. When ϕ_k increases to a certain threshold, the optimization effect for time consumption slows down gradually.

It is worth noting that the gain of ϕ_k to total time optimization is far less than K.

Therefore, MEC-NBS can effectively reduce the system delay, in which the combination of data caching and computational offloading than one separate computational offloading brings more delay benefits.

5.5. Different Data Transmission Rates between C and MEC-NBS

As shown in Figure 8, we measured the total time consumption by the system in completing all tasks under the 3 schemes with different data transmission rates between C and MEC-NBS. In simulations, we increase the $s_k^{\text{C2NBS}}, \forall k \in \mathcal{K}$ from 50 to 90 MB/s, while other settings are listed in Table 1, except fixed values $M = 60$, $R = 6$, $K = 6$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{NBS2C}} = 30$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$.

The data transmission rates between C and MEC-NBS impact the time consumptions of transmission processes in further offloading and data caching. As the transmission rates increase, the time consumptions decrease, so the time benefits gained by further offloading and data caching grows. As shown in Figure 8, the curves of our scheme and OMEC appear a downward trend as $s_k^{\text{C2NBS}}, \forall k \in \mathcal{K}$ increase, whereas the curve of NMEC is unchanged since further offloading and data caching are disabled.

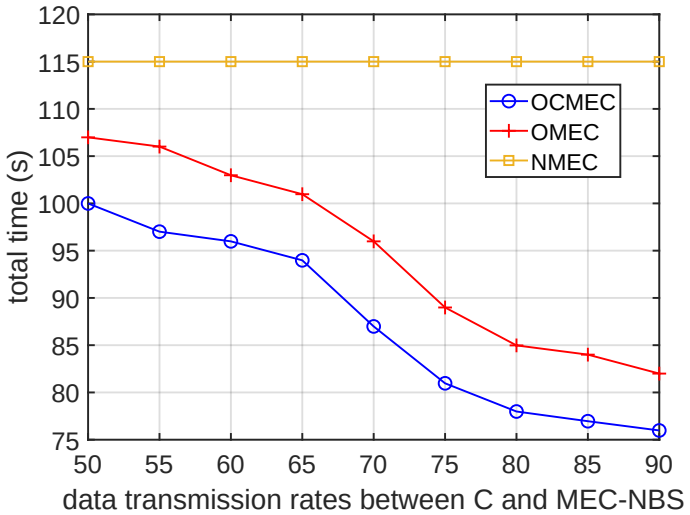

Figure 8. f gained by the 3 schemes with the increase of s_k^{C2NBS}, the data transmission rates between C and MEC-NBS, $k = 1, 2, \ldots, K$.

5.6. Different Data Transmission Rates between MEC-NBS and C

As shown in Figure 9, we measured the total time consumption by the system in completing all tasks under the 3 schemes with different data transmission rates between C and MEC-NBS. In simulations, we increase the $s_k^{\text{NBS2C}}, \forall k \in \mathcal{K}$ from 10 to 40 MB/s, while other settings are listed in Table 1, except fixed values $M = 60$, $R = 6$, $K = 6$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$.

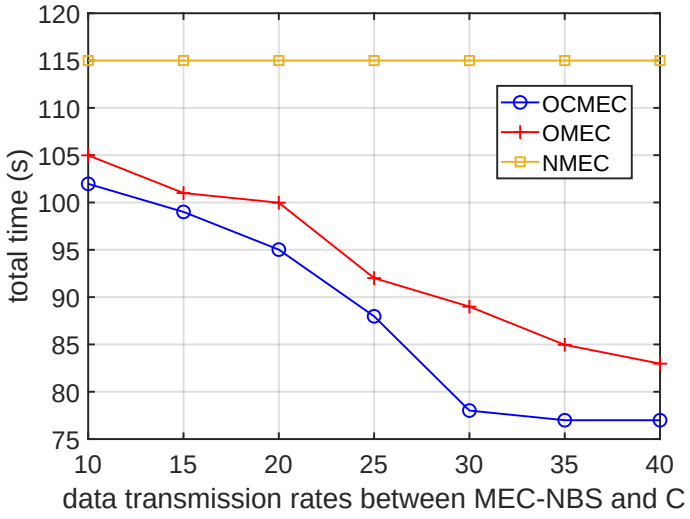

Figure 9. f gained by the 3 schemes with the increase of s_k^{NBS2C}, the data transmission rates between MEC-NBS and C, $k = 1, 2, \ldots, K$.

Similar to the data transmission rates between C and MEC-NBS, $s_k^{\text{NBS2C}}, \forall k \in \mathcal{K}$ impacts the time consumptions of transmission processes in further offloading and data caching. Although the curves of our scheme and OMEC are not very smooth, we can still notice that the overall trend of the curves are decreasing. Similarly, the curve of NMEC is still unchanged because further offloading and data caching are disabled.

5.7. Different Data Transmission Rates between MEC-BS and MEC-NBS

As shown in Figure 10, we measured the total time consumption by the system in completing all tasks under the 3 schemes with different data transmission rates between MEC-BS and MEC-NBS. In simulations, we increase the s_k^{BS2NBS} and $s_k^{\text{NBS2BS}}, \forall k \in \mathcal{K}$ from 10 to 40 MB/s, while other settings are listed in Table 1, except fixed values $M = 60$, $R = 6$, $K = 6$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{NBS2C}} = 30$ MB/s, $\forall k \in \mathcal{K}$.

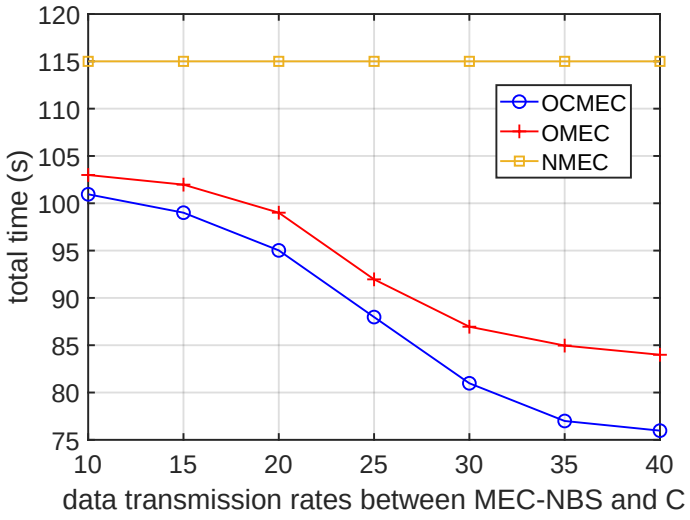

Figure 10. f gained by the 3 schemes with the increase of s_k^{BS2NBS} and s_k^{NBS2BS}, the data transmission rates between MEC-BS and MEC-NBS, $k = 1, 2, \ldots, K$.

Similar to the data transmission rates between MEC-NBS and C, s_k^{BS2NBS} and s_k^{NBS2BS}, $\forall k \in \mathcal{K}$ impacts the time consumptions of transmission processes in further offloading and data caching. The overall trend of the curves of OCMEC and OMEC are decreasing. Similarly, the curve of NMEC is still unchanged because further offloading and data caching are disabled.

5.8. Comparison with Other Optimization Algorithms

As shown in Figure 11, we compare our algorithm with two other ones, including the Particle Swarm Optimization (PSO) algorithm and the Simulated Annealing (SA) algorithm. The convergence speeds of the 3 algorithms are measured to find the solution of our optimization problem with $M = 70$, $R = 6$, $K = 5$, $\phi_k = 2.4 \times 10^6$ MBPS, $s_k^{\text{C2NBS}} = 80$ MB/s, $s_k^{\text{NBS2C}} = 30$ MB/s, $s_k^{\text{BS2NBS}} = s_k^{\text{NBS2BS}} = 30$ MB/s, $\forall k \in \mathcal{K}$, while other settings are listed in Table 1. The simulation results show that although all of the 3 algorithm can converge after multiple iterations, our Genetic algorithm has the fastest convergence speed (our Genetic algorithm is converged at iteration 252, PSO algorithm is converged at iteration 281, and SA algorithm is converged at iteration 297). It demonstrates that our algorithm is more suitable for our optimization problem.

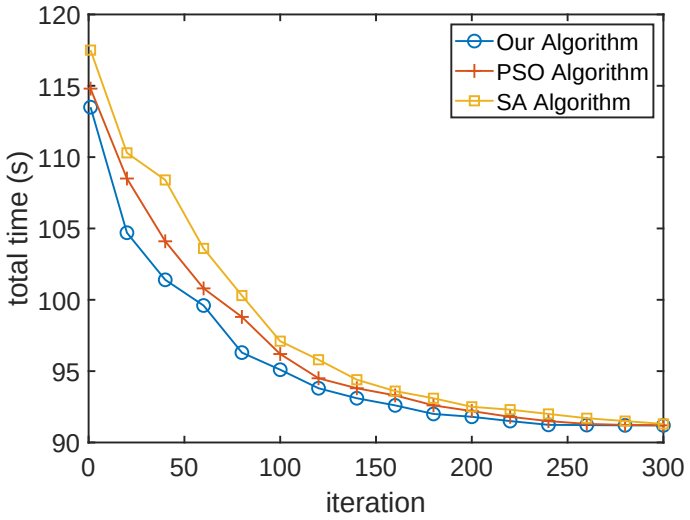

Figure 11. Convergence analysis of the 3 algorithms.

6. Conclusions

As one of the key research directions in the future mobile network, mobile edge computing technology can effectively alleviate the computing pressure and data interaction pressure of mobile cloud, reduce the response delay of mobile terminals, and improve the overall efficiency of the system.

In this paper, the characteristics of computing offloading and data caching in mobile edge computing are deeply studied, and a new resource management scheme based on the cooperation of multiple MEC base stations is proposed. In our scheme, MEC-BS and MEC-NBS can perform computational tasks offloaded from the cloud and cache the data needed to compute tasks from the cloud, minimizing the time consumed to perform all tasks.

In our research, various computing tasks of mobile terminals are defined by computing load and data load, and resource management problem is transformed into optimization problem with the objective of minimizing system time consumption. The Cooperative Resource Management Algorithm proposed in this paper consists of several separate and parallel running cycles. The performance of our scheme is evaluated and compared with the following two schemes: (1) schemes without data caching function; and (2) schemes without data caching and computing offloading. Compared with the other two schemes, our scheme shows obvious superiority in three different situations, and greatly improves the performance of the system.

Our scheme provides a new resource management method leveraging the cooperation of multiple MEC-BS. It can efficiently enhance the performance of computation-intensive and delay-sensitive applications, and improve the user experience and device supports on them.

In our future works, we plan to extend our research to the area of IoT applications, such Internet of Vehicles, Industrial Internet of Things, etc. The joint computation offloading

and data caching problem will be studied to efficiently exploit the joint utilization of the resources of all BSs and the cloud to enhance the performance of task processing in these applications.

Author Contributions: Conceptualization, T.L., W.F., F.W., W.X. and W.Y.; methodology, T.L., W.F., F.W., W.X. and W.Y.; software, T.L., W.F., F.W., W.X. and W.Y.; validation, T.L., W.F., F.W., W.X. and W.Y.; formal analysis, T.L., W.F., F.W., W.X. and W.Y.; investigation, T.L., W.F., F.W. and W.X.; data curation, T.L., W.F., F.W. and W.X.; writing—original draft preparation, T.L., W.F., F.W. and W.X.; writing—review and editing, T.L., W.F. and W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. Mobile edge computing: A key technology towards 5G. *ETSI White Pap.* **2015**, *11*, 1–16.
2. Ahmed, A.; Ahmed, E. A survey on mobile edge computing. In Proceedings of the 2016 10th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 7–8 January 2016; pp. 1–8.
3. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2322–2358. [CrossRef]
4. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. [CrossRef]
5. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile Edge Computing: A Survey. *IEEE Internet Things J.* **2017**, *5*, 450–465. [CrossRef]
6. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Architecture & Orchestration. *IEEE Commun. Surv. Tutorials* **2017**. [CrossRef]
7. Xu, X.; Zhang, X.; Gao, H.; Xue, Y.; Qi, L.; Dou, W. BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing. *IEEE Trans. Ind. Informatics* **2019**, *16*, 4187–4195. [CrossRef]
8. Bista, B.B.; Wang, J.; Takata, T. A Probabilistic Offloading Approach in Mobile Edge Computing. In Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, Antwerp, Belgium, 7–9 November 2019; Springer: Berlin, Germany, 2019; pp. 266–278.
9. Gu, H.; Wang, H. A Distributed Caching Scheme Using Non-Cooperative Game for Mobile Edge Networks. *IEEE Access* **2020**, *8*, 142747–142757. [CrossRef]
10. Liu, Y.; He, Q.; Zheng, D.; Xia, X.; Chen, F.; Zhang, B. Data caching optimization in the edge computing environment. *IEEE Trans. Serv. Comput.* **2020**. [CrossRef]
11. Kobayashi, R.; Adachi, K. Radio and computing resource allocation for minimizing total processing completion time in mobile edge computing. *IEEE Access* **2019**, *7*, 141119–141132. [CrossRef]
12. Wang, C.; Feng, D.; Zhang, S.; Chen, Q. Video Caching and Transcoding in Wireless Cellular Networks With Mobile Edge Computing: A Robust Approach. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9234–9238. [CrossRef]
13. Mao, Y.; Zhang, J.; Song, S.H.; Letaief, K.B. Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 5994–6009. [CrossRef]
14. Mao, Y.; Zhang, J.; Letaief, K.B. Joint Task Offloading Scheduling and Transmit Power Allocation for Mobile-Edge Computing Systems. In Proceedings of the Wireless Communications and NETWORKING Conference, San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
15. Yang, G.; Hou, L.; He, X.; He, D.; Chan, S.; Guizani, M. Offloading Time Optimization via Markov Decision Process in Mobile Edge Computing. *IEEE Internet Things J.* **2020**, *8*, 2483–2493. [CrossRef]
16. Kuang, Z.; Li, L.; Gao, J.; Zhao, L.; Liu, A. Partial offloading scheduling and power allocation for mobile edge computing systems. *IEEE Internet Things J.* **2019**, *6*, 6774–6785. [CrossRef]
17. Liang, B.; Fan, R.; Hu, H.; Zhang, Y.; Zhang, N.; Anpalagan, A. Nonlinear Pricing Based Distributed Offloading in Multi-User Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2020**, *70*, 1077–1082. [CrossRef]
18. Pham, Q.V.; Le, L.B.; Chung, S.H.; Hwang, W.J. Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation. *IEEE Access* **2019**, *7*, 16444–16459. [CrossRef]

19. Fan, W.; Liu, Y.; Tang, B. Adaptive Application Component Mapping for Parallel Computation Offloading in Variable Environments. *Ksii Trans. Internet Inf. Syst.* **2015**, *9*, 4347–4366.
20. Cui, Y.; Zhang, D.; Zhang, T.; Chen, L.; Piao, M.; Zhu, H. Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices. *AEU Int. J. Electron. Commun.* **2020**, *118*, 153134. [CrossRef]
21. Eliodorou, M.; Psomas, C.; Krikidis, I.; Socratous, S. Energy efficiency for MEC offloading with NOMA through coalitional games. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
22. Zhou, S.; Jadoon, W. Jointly Optimizing Offloading Decision and Bandwidth Allocation with Energy Constraint in Mobile Edge Computing Environment. *Computing* **2021**. [CrossRef]
23. Song, Z.; Liu, Y.; Sun, X. Joint Task Offloading and Resource Allocation for NOMA-Enabled Multi-Access Mobile Edge Computing. *IEEE Trans. Commun.* **2020**, *69*, 1548–1564. [CrossRef]
24. You, C.; Huang, K.; Chae, H.; Kim, B.H. Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 1397–1411. [CrossRef]
25. Wang, J.; Feng, D.; Zhang, S.; Liu, A.; Xia, X.G. Joint Computation Offloading and Resource Allocation for MEC-enabled IoT Systems with Imperfect CSI. *IEEE Internet Things J.* **2020**, *8*, 3462–3475. [CrossRef]
26. Sardellitti, S.; Scutari, G.; Barbarossa, S. Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Trans. Signal Inf. Process. Over Netw.* **2015**, *1*, 89–103. [CrossRef]
27. Liu, L.; Sun, B.; Tan, X.; Xiao, Y.S.; Tsang, D.H. Energy-efficient Resource Allocation and Channel Assignment for NOMA-based Mobile Edge Computing. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, 15–18 April 2019; pp. 1–6.
28. Wang, C.; Yu, F.R.; Liang, C.; Chen, Q.; Tang, L. Joint Computation Offloading and Interference Management in Wireless Cellular Networks with Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2017**, *66*, 7432–7445. [CrossRef]
29. Song, F.; Xing, H.; Luo, S.; Zhan, D.; Dai, P.; Qu, R. A Multiobjective Computation Offloading Algorithm for Mobile-Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 8780–8799. [CrossRef]
30. Shekhar, C.A.; Sharvani, G. MTLBP: A Novel Framework to Assess Multi-Tenant Load Balance in Cloud Computing for Cost-Effective Resource Allocation. *Wirel. Pers. Commun.* **2021**. [CrossRef]
31. Yang, J.; Xiang, Z.; Mou, L.; Liu, S. Multimedia resource allocation strategy of wireless sensor networks using distributed heuristic algorithm in cloud computing environment. *Multimed. Tools Appl.* **2020**, *79*, 35353–35367. [CrossRef]
32. Gharehpasha, S.; Masdari, M. A discrete chaotic multi-objective SCA-ALO optimization algorithm for an optimal virtual machine placement in cloud data center. *J. Ambient. Intell. Humaniz. Comput.* **2020**. [CrossRef]
33. Kong, L.; Mapetu, J.P.B.; Chen, Z. Heuristic load balancing based zero imbalance mechanism in cloud computing. *J. Grid Comput.* **2020**, *18*, 123–148. [CrossRef]
34. Satria, D.; Park, D.; Jo, M. Recovery for overloaded mobile edge computing. *Future Gener. Comput. Syst.* **2016**, *70*, 138–147. [CrossRef]
35. Gross, D. *Fundamentals of Queueing Theory*; John Wiley & Sons: Hoboken, NJ, USA, 2008.