Multicast-Aware Optimization for Resource Allocation with Edge Computing and Caching

Hao Hao[a], Changqiao Xu[a,*], Shujie Yang[a], Lujie Zhong[b], Gabriel-Miro Muntean[c]

[a] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, P. R. China.*
[b] *Information Engineering College, Capital Normal University, Beijing 100048, P. R. China.*
[c] *Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Dublin, Ireland.*

Abstract

Mobile edge computing (MEC) can save backhaul network bandwidth and meet latency requirements, rising as a key technology for computation-intensive and delay-sensitive applications. Although many works have studied various problems of MEC (e.g. computation offloading, resource allocation), most of them only consider unicast scenarios, and ignore multicast issues. The reason is that MEC aiming at personalized computing of users conflicts with multicast which demands transmission of the same data stream. This makes MEC and multicast seem inconsistent. However, in fact, there will be lots of services, whose computation process is different but the result may be same (e.g. media push), which can greatly benefit from multicast over MEC and make the combination of multicast and MEC meaningful. In this paper, we first consider challenges and propose multicast-aware resource allocation for MEC, which jointly optimizes computing and caching in multicast scenarios. Consider the complexity, we formulate the problem as an online optimization by jointly minimizing the average time delay and energy consumption, and separate it into two subproblems that can be solved per time slot. Then, an efficient online algorithm called MA-ECC is proposed to solve the problem. Finally, numerical results show that MA-ECC has fast convergence and can effectively reduce service latency while still keeping energy consumption low.

Keywords: Mobile Edge Computing, Multicast, Stochastic Optimization, Resource Allocation.

1. Introduction

Many new computational-demanding services, such as video codec, face detection, etc. are emerging and becoming essential elements in our life. While enriching our daily life, they also causes more consumption of computing, energy and other resources for mobile devices. To address this issue, researchers proposed cloud computing which can effectively reduce the computing pressure of mobile devices by offloading computing services to remote cloud servers.[1] However, cloud computing tends to have high service delay because of the long transmission from users to cloud servers, and it is hard for the cloud servers to support computing demand of all devices. In this context, mobile edge computing (MEC) [2] was proposed to achieve lower service delay by moving computing tasks from the remote servers to the edge base stations which are closer to users.

An important avenue for research in MEC is computing resource allocation in heterogeneous cellular networks [3][4][5][6][7][8]. The reason is that unlike the cloud, computing resource of base stations(BSs) is limited, implying that computing resource allocation affects significantly MEC efficiency. In addition, BSs need to cache related databases/libraries before providing edge computing [9][10][11][12][13]. Consequently, resource allocation with joint optimization of computing and caching is of double importance for MEC.

Although some excellent works have focused on computing resource allocation or service caching, they have usually optimized one these aspects only. Moreover, existing works studied computing and caching resource allocation in unicast scenarios, but have not focused on multicast yet. Multicast technology, which effectively utilizes the intrinsic broadcast nature of BS channels, is an efficient way to deliver identical content to satisfy multiple requests [14]. At first glance, MEC and multicast seem to be two orthogonal research directions, focusing on personalized processing and same content transmission, respectively. However, we discover that many MEC services can benefit from multicast.

For example, in live video case, the applications analyse packet loss, available bandwidth and other factors to provide appropriate bitrate version for users by video codec. Considering that the number of video versions (e.g. 360P, 720P, 1080p, etc.) is limited, users with similar link qualities will get the same bitrate version [15]. Therefore, we can encode at edge base stations and transmit the same version to users with similar link qualities by multicast to improve transmission efficiency, which is an effective combination of MEC and multicast. Another relevant example involves media push. A typical solution is that users with similar interests are allocated to the same group and are pushed the same content [16][17]. If we can joint

*Corresponding author: Changqiao Xu
Email addresses: hao_hao@bupt.edu.cn (Hao Hao), cqxu@bupt.edu.cn (Changqiao Xu), sjyang@bupt.edu.cn (Shujie Yang), zljict@gmail.com (Lujie Zhong), gabriel.muntean@dcu.ie. (Gabriel-Miro Muntean)

MEC and multicast, edge base stations analyze user interests and allocate users with similar interests to the same group, then use multicast to push same contents to users in same group, which will greatly improve transmission efficiency. Extrapolating these examples, note that these services have different computation processes, but may result in the same outcome, which means multicast is of high potential in MEC. Therefore, it is valuable to study the resource allocation with edge computing in multicast scenarios.

The multicast-aware resource allocation with joint optimization of computing and caching faces many challenges. First, computing, caching and multicast are highly coupled and interacting with each other, so they should be addressed jointly. There are two variables in this problem, caching decision which is a zero-one type integer, and computing allocation decision which is a continuous variable. The problem is hard to be solved because of a mixture of discrete variable and continuous variable, which makes some methods such as reinforcement learning inappropriate.

Second, the collaboration between edge base stations and cloud servers should be considered. As we all know, the limited resources of the edge base station make it impractical to provide all services. In order to provide high quality services, cloud servers are needed besides edge base stations. But, different services require different aspects of resources [18]; for instance face recognition focuses on computing resources, but VR needs more storage space. Therefore, how to coordinate the tasks allocation between edge and cloud is a big challenge.

Third, network variants are highly dynamic and stochastic. We should use long-term average performance to evaluate an algorithm. However, this usually needs future information which is hard to be obtained in dynamic network. So, it is very challenging to optimize long-term average performance.

In this paper, we addresses multicast-aware resource allocation with joint optimization of computing and caching. The Multicast-Aware Caching and Computing algorithm (MA-ECC) is proposed to enable efficient allocation of resources. The main contributions of this paper are as follows:

(1) We discover the link between edge computing and multicast, and innovatively propose the multicast-aware resource allocation problem , which is formulated as minimization of service latency under energy consumption constraints.

(2) Due to the lack of future information, we transform original problem into per time-slot optimization. Considering the mixture of discrete variable and continuous variable, the problem is further decomposed into two subproblems: caching decision and computing resource allocation. Then, we solve them by employing *implicit enumeration method* and *Karush-Kuhn-Tucher (KKT)* solution, respectively. Finally, a novel efficient algorithm, MA-ECC, is proposed to solve the joint optimization problem.

(3) Furthermore, we prove the performance of MA-ECC theoretically and design extensive simulation experiments to verify it. Results show that MA-ECC performs well in terms of both service latency and energy consumption.

The paper is organized as follows. The related works are reviewed in Section II. The system model is introduced in Section III. We formulate the problem in Section IV. Section V decomposes the problem into two subproblem and presents the designed MA-ECC algorithm. Section VI discusses the simulation results. Finally, Section VII is the conclusion.

2. Related Work

Avoiding long-distance transmission from users to cloud servers, MEC can effectively reduce the service delay and relieve the backhaul link pressure, which is becoming an important computing paradigm. Many scholars have conducted research on MEC. The authors of [3] transform the problem into a distributed convex optimization and solve it based on alternating direction method. [4] designs a decentralized computation offloading algorithm by game theoretical and proves the upper bound of algorithm. In [5], authors study the MEC in Device-to-Device scenario. They propose an integrated framework for computation offloading and interference management to optimize service delay. Considering the limited power of mobile devices, [6] formulate the computation offloading to a mixed-integer problem and reduced the energy consumption of communication in MEC. In [7], author propose an energy-efficient computation offloading strategy, which contains three step of computation offloading selection, clock frequency control and transmission power allocation, to shorten the service delay while reducing energy consumption. Considering the stochastic properties of service requests, the authors of [8] study the stochastic resource allocation strategy by deep reinforcement learning. These works are focus on the computation offloading, but does not give the computing resources allocation scheme for each service.

In terms of edge caching, authors [9] investigate the content caching problem for the adaptive streaming. The caching management in mobile network scenario is studied in [10]. They propose a mathematical framework which considers content request characteristics and content catalogs to reduce delivery delay. Authors of [11] provide a cooperative video caching mechanism by jointly considering users' request similarity, users' movement behavior and users' demand, which greatly improves quality of experience. In [12], authors improve the hit ratio by predicting the popularity of contents. In [13], authors improve the hit ratio by proposing a cache prefetching strategy. They use Bayesian network theory to select contents. These works mainly focus on content caching scheme and not consider the joint optimization of caching and computation.

To enhance video rate adaptation, a joint optimization of computing and caching is proposed in software-defined networks [19]. In [20], authors research a joint caching and computing problem for adaptive bitrate (ABR) delivery, which considers the constraints of both storage space and computing capacity. The most related work is [21]. Authors formulate the joint optimization of content caching, computation offloading and computing resource allocation as an mixed integer nonlinear programming, and solve it by generalized benders decomposition method.But there are several differences to our work.

First, while unicast scenario is considered in [21], we focus on a multicast scenario. Second, while only the computing delay is optimized in [21], we optimize the service latency which contains computing delay and transmission time. Third, while the computing tasks are assumed to be divisible in [21], we consider they are the smallest processing unit.

3. System Model

This section proposes the system model for multicast-aware joint edge computing and caching resource allocation. The scenario is described first and then the request model, communication model, caching model and computation model are introduced. Table 1 lists the mathematical notations.

Table 1: Mathematical notations

Notation	Explanation
$\mathcal{B}$	The set of base stations
C_n	Storage space of b_n
F_n	Computing capacity (maximum frequency) of base station b_n
H_n	Multicast channel gain of b_n
P_n	Multicast power of b_n
σ_n	Noise power of b_n
e_n	Unit energy consumption of b_n
ζ_n	Effective switched capacitance of b_n
$\mathcal{K}$	The set of services
m_k	Transmission data size of service k
c_k	Storage space requirement of service k
d_k	Computing resource requirement of service k
$f(t)$	Computation resource allocation vector
$x(t)$	Caching resource allocation vector
E_n^r	The limited power of b_n
$T_{n,k}^M(t)$	Time delay of b_n for transmit service k
$E_{n,k}^M(t)$	Energy consumption of base station b_n for transmit service k
$E_{n,k}^S(t)$	Energy consumption of base station b_n for caching service k
$T_{n,k}^C(t)$	Computation time of b_n for service k
$E_{n,k}^C(t)$	Energy consumption of base station b_n for computing service k
$R_{n,k}(t)$	The request to b_n for service k
$S_{n,k}(t)$	The actual service volume of k in b_n
$H_n(t)$	Length of energy queues

3.1. Scenario Description

A heterogeneous cellular network containing a MBS and N SBSs is considered, as shown in Fig. 1. Each BS is equipped with computing capabilities and storage space to provide computing services. SBS coverage areas may be joint, but user access is not the scope of this paper. Therefore, we assume users only request to one SBS, as presented in literature [22][23].

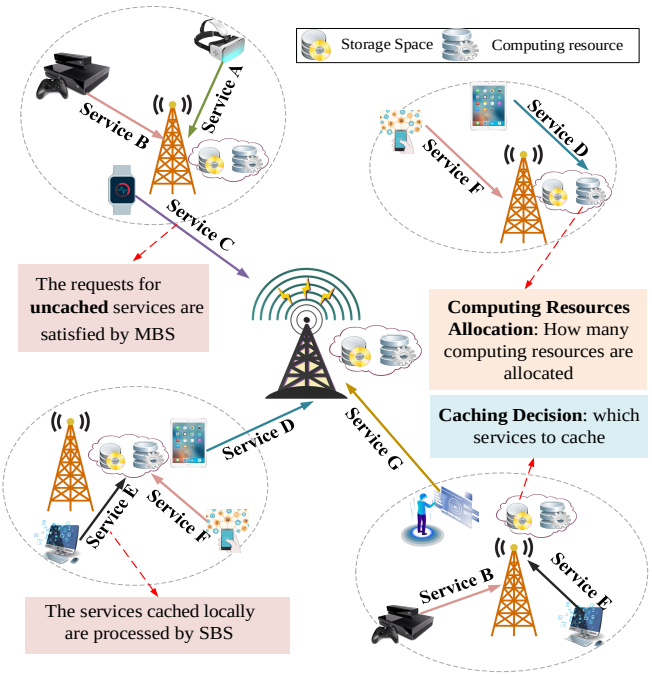

Figure 1: System architecture

MBS can serve all SBSs and users while the coverage areas of SBS is limited. We denote the set of base stations $\mathcal{B} \triangleq \{b_0, b_1, ..., b_N\}$ where MBS is represented by b_0 and we denote $\mathcal{B}^+ \triangleq \{b_1, b_2, ..., b_N\}$ as the set of SBSs. Each base station b_n is associated with storage space C_n and computing capacity F_n (e.g. the maximum frequency of CPU). The storage space C_n is used to cache services data (e.g. databases/libraries and content) and the computing capability F_n is used for support of diverse computation processes of services. Similar to reality, MBS which is regared as the cloud servers can store all services data and has strong computing capability [24]. Different from MBS, the storage space of SBSs which are edge nodes is limited and can only store part of services.

There are K independent services, expressed by the set $\mathcal{K} \triangleq \{1, 2, ..., K\}$. Every service k has three important attributes (c_k, d_k, m_k) where c_k is the storage space required for caching service data (e.g. databases/libraries and contents) of k, d_k denotes the average computation required to complete service k, i.e. the number of CPU cycles, and m_k is the data size of results when the computing of service k is completed. For example, in media pushing case, c_k is the data size of related databases/libraries and content, d_k is the computation required for analysing user interests, and m_k is the size of media that is recommended to users. Moreover, considering each service is indivisible (i.e. it is difficult to get accurate user interests if divide user's request record into several parts and analyse them separately), we assume that each logically independent service is the smallest processing unit. For a single service, mobile users first request it from the SBS which they are associated to. The SBS will allocate computing resources to complete the

service if it caches the related data. Otherwise, the SBS will offload the requests to MBS and MBS will provide the service.

Without loss of general assumption, the time is slotted [25], i.e., $\mathcal{T} = \{0, 1, ..., T-1\}$. In each time slot t, BSs provide the services which have been requested according to multicast. Due to the difference of storage space and computing resource between MBS and SBSs, for a service, the caching status and allocated computing resource will affect services latency and energy consumption. Therefore, we consider the following multicast-aware edge caching and computing problem with two sub-problems:

1. Which service to cache? The decision to make is which services data should be cached in SBS due to the limited storage space.

2. How to allocate computing resources? This refers to how many computing resources are allocated to each service with the limited computing capability of SBS.

In this paper, we optimize the problem by minimizing overall latency of services with the constraints of energy consumption. Next, system models for request, communication, caching and computation are presented in details.

3.2. Request Model

A user sends service requests to the associated base station. The process of service requests is two-tier as shown in Fig.2. First, users request service k from the SBS they are associated to. If the SBS stores related data(e.g. database/libraries and contents), it will allocate computing capabilities to process the task and provide related service. Otherwise, the SBS forwards the request to MBS and MBS assigns pre-fixed computing resource [26] to it. In other word, for SBS b_n, it will forward the requests of service k to MBS if not stores related data. It's worth noting that the service delay of MBS is much longer than the edge computation time because of long distance transmission. Denote $\boldsymbol{x} \triangleq (x_{n,k}(t) \in \{0,1\} : n \in \mathcal{B}, k \in \mathcal{K}, t \in \mathcal{T})$ as the caching variable, i.e., $x_{n,k}(t) = 1$ if base station b_n stores data of service k at time slot t and $x_{n,k}(t) = 0$ otherwise. The actual volume of service k to b_n is formulated as:

$$S_{n,k}(t) = R_{n,k}(t) x_{n,k}(t) \tag{1}$$

where $R_{n,k}(t)$ is the number of requests to b_n for service k at time slot t.

For MBS b_0, alongside requests from users, it also needs to satisfy the requests that are forwarded from SBSs. Therefore, at time slot t, the actual service volume is the sum of the two components. This is formulated as:

$$S_{0,k}(t) = R_{0,k}(t) + \sum_{n \in \mathcal{B}^+} R_{n,k}(t)[1 - x_{n,k}(t)] \tag{2}$$

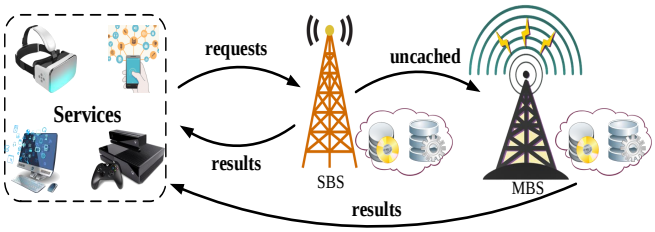

Figure 2: The process of service requests

3.3. Communication Model

Based on multicast communication mechanism, we build the communication model considering transmission time and energy consumption. Assuming that the spectrum between SBSs and MBS is orthogonal. For base station b_n, $H_n(t)$ is the multicast channel gain and $P_n(t)$ is the multicast power. Then the multicast data rate of b_n is as follows:

$$r_n(t) = B_n(t) \log_2(1 + \frac{P_n(t) H_n(t)}{\sigma_n^2(t)}) \tag{3}$$

where $\sigma_n^2(t)$ denotes the noise power, and $B_n(t)$ represents the channel bandwidth. Although b_n provides services to $S_{n,k}(t)$ users, the data transfer volume is m_k instead of $m_k S_{n,k}(t)$ because of the multicast technology. Therefore, for base station b_n, the transmission time of service k is:

$$T_{n,k}^M(t) = \frac{m_k}{r_n(t)} \tag{4}$$

where m_k is the result size of service k after computing. The energy consumption of multicast is represented as follows:

$$E_{n,k}^M(t) = P_n(t) T_{n,k}^M(t) \tag{5}$$

3.4. Caching Model

In this subsection, we present the caching model considering storage space constraint and energy consumption. MBS has enough storage space and can cache all services data which means $\boldsymbol{x_0(t)} = \mathbf{1}$. For SBSs, requests for service k can be satisfied locally if SBSs cache related data, reducing transmission time and improving service quality. However, because of the constraint of caching capacity, SBSs can not simultaneously cache all services data. Therefore, it is important for SBS b_n to decide efficiently which services to store. Moreover, considering the limited storage space, the caching decision of SBS b_n should satisfy the following constraint:

$$\sum_{k \in \mathcal{K}} c_k x_{n,k}(t) \leq C_n \tag{6}$$

The energy consumption of b_n associated to caching service k is obtained as follows:

$$E_{n,k}^S(t) = c_k x_{n,k}(t) e_n \tag{7}$$

where e_n is the unit energy consumption, considered a fixed value.

3.5. Computation Model

It is obvious that different computing resource allocation decisions lead to different computation times and energy consumption. Let $\boldsymbol{f} \triangleq (f_{n,k}(t): n \in \mathcal{B}, k \in \mathcal{K}, t \in \mathcal{T})$ denote the allocation decision, which means that base station b_n allocates $f_{n,k}(t)$ computing resource to complete service k in time slot t. Similar to caching, the computing resource allocation decision for SBSs is subject to the limited computing resource:

$$\sum_{k \in \mathcal{K}} f_{n,k}(t) \leq F_n \tag{8}$$

As mentioned above, the computing process of services is indivisible and the process of service request is two-tier. SBSs will allocate computing resources for services k if they cache related data. Otherwise, MBS assigns computing resource to the service. For service k, the computation demand of SBSs in time slot t is presented uniformly as $d_k S_{n,k}(t)$, which is different from the calculation method of data transfer volume in the communication model. The reason is that different users have different inputs leading to different computations. Therefore, the computation time $T^C_{n,k}(t)$ of SBS b_n for service k can be expressed as:

$$T^C_{n,k}(t) = \frac{d_k S_{n,k}(t)}{f_{n,k}(t)} \tag{9}$$

For MBS, the calculation of computation time is different from SBS because the MBS allocates a pre-fixed computing resource to every request, regardless of the actual service volume. Therefore, the computation time for service k is:

$$T^C_{0,k}(t) = \frac{d_k}{f_{0,k}(t)} \tag{10}$$

The energy that each CPU cycle consumes is $\zeta_n(f^2_{n,k}(t))$, where ζ_n is the energy coefficient relevant to chip architecture [27]. The energy consumption of b_n for service k is defined as follows:

$$E^C_{n,k}(t) = d_k S_{n,k}(t) \zeta_n(f^2_{n,k}(t)) \tag{11}$$

4. PROBLEM FORMULATION AND TRANSFORMATION

In this section, based on the already-introduced system model, we first formulate the multicast-aware joint resource allocation as an optimization problem to minimize service latency while keeping energy consumption low. Considering the lack of future information and intractability of the problem, we further transform the problem into an online solvable problem.

4.1. Problem Formulation

For each request, the service latency of k consists of two parts: transmission time and computation time. Therefore, the total latency of base station b_n for all requests is:

$$T_n(t) = \sum_{k \in \mathcal{K}} (T^M_{n,k}(t) + T^C_{n,k}(t)) S_{n,k}(t) \tag{12}$$

To simplify the total latency for SBS, we find that $S^2_{n,k}(t) = S_{n,k}(t) R_{n,k}(t)$ because of $x^2_{n,k}(t) = x_{n,k}(t)$ and $S^2_{n,k}(t) = R_{n,k}(t) x_{n,k}(t)$. Therefore, for SBS, we can replace the quadratic term $x^2_{n,k}(t)$ and obtain the total latency as:

$$T_n(t) = \sum_{k \in \mathcal{K}} (T^M_{n,k}(t) S_{n,k}(t) + T^C_{n,k}(t) R_{n,k}(t)) \tag{13}$$

The overall energy consumption is due to multicast consumption, caching consumption and computation consumption. As we all know, base stations provide services only when the related computation is complete. In other word, only the services that are allocated computing resources can be provided and have transmission energy consumption. So the total energy consumption of SBS b_n for all services can be expressed as:

$$E_n(t) = \sum_{k \in \mathcal{K}} \{E^M_{n,k}(t) \mathbf{1}_{\{f_{n,k}(t) \neq 0\}} + E^C_{n,k}(t) + E^S_{n,k}(t)\} \tag{14}$$

where $\mathbf{1}_{\{x\}} = 1$ if x is true, otherwise $\mathbf{1}_{\{x\}} = 0$.

The objective of multicast-aware joint resource allocation problem is to minimize overall service latency with limited energy consumption. We formulate this problem as follows:

$$\min_{\boldsymbol{x}, \boldsymbol{f}} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n \in \mathcal{B}} T_n(t)$$

$$s.t. \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} E_n(t) \leq E^r_n, \forall n \in \mathcal{B}^+ \tag{15a}$$

$$\sum_{k \in \mathcal{K}} c_k x_{n,k}(t) \leq C_n, \forall n \in \mathcal{B}^+ \tag{15b}$$

$$\sum_{k \in \mathcal{K}} f_{n,k}(t) \leq F_n, \forall n \in \mathcal{B}^+ \tag{15c}$$

$$\frac{f_{n,k}(t)}{F_n} \leq x_{n,k}(t), \forall n \in \mathcal{B}^+, \forall k \in \mathcal{K} \tag{15d}$$

$$x_{n,k} \in \{0, 1\} \tag{15e}$$

The first constraint (15a) guarantees that long-term average energy consumption of SBS b_n is limited by power E^r_n. Constraint (15b) ensures the storage space allocated to all services is limited by the caching capacity of SBS. Constraint (15c) indicates the computing capacity of SBS is limited. The fourth constraint (15d) means that we can only allocate computing resource to the services whose related data has been stored in BS. The constraint (15e) is due to the fact that caching variable is binary.

It is intractable to derive the optimal solution of the above problem. The first challenge is lacking of future information. It is difficult and impractical to solve this problem because it requires the request status of users in all time slots. Moreover, the problem is a mixed integer nonlinear programming with discrete variable and continuous variable, which is still challenging even if the request status is known a priori. Therefore, it is imperative to transform this problem into an online solvable problem without the need for predicting future information.

4.2. Problem Transformation

In this subsection, we transform the problem (15) into per time-slot optimization which does not require future information. We construct energy queues to satisfy the long-term energy constraint (15a):

$$H_n(t+1) = max[H_n(t) - E_n^r + E_n(t), 0] \tag{16}$$

where $H_n(0) = 0$. By its dynamic, $H_n(t)$ indicates the backlog of deviation of current energy consumption from the energy constraint.

Lemma 1. *If $H_n(t)$ is mean rate stable, our desired average energy consumption constraint (15a) is satisfied. We define the mean rate stable as:*

$$\lim_{t\to\infty} \frac{\mathbb{E}\{H_n(t)\}}{t} = 0 \tag{17}$$

where $\mathbb{E}\{.\}$ denotes the expectation.

The proof of **Lemma 1** is shown in Appendix A. □

To represent the "congestion level" in energy queues, we define the energy function as:

$$q(t) \triangleq \frac{1}{2}\sum_{n\in\mathcal{N}}(H_n^2(t)) \tag{18}$$

The energy function represents the status of energy queues. A small $q(t)$ means queues are highly stable, which implies the energy deficit is small. We persistently reduce the value of energy function to stabilize the energy queues and satisfy the energy consumption constraints. In addition, we introduce the drift which is the variation of energy function:

$$\Delta q(t) \triangleq \mathbb{E}\{q(t+1) - q(t)|H(t)\} \tag{19}$$

Energy queues become more stable if $\Delta q(t)$ is smaller. Therefore, in order to jointly optimize the service latency and queue stability, the drift-plus-penalty is defined as:

$$\Theta(t) \triangleq \Delta q(t) + \mathbb{E}\{V\sum_{n\in\mathcal{B}} T_n(t)\} \tag{20}$$

where V is a positive parameter to emphasize the significance of service latency. Considering the inequality $(max[a-b+c,0])^2 \leq a^2+b^2+c^2+2a(c-b)$, we further find the upper-bound of energy function as follows:

$$\begin{aligned}\Delta q(t) &\leq \frac{1}{2}\mathbb{E}\{\sum_{n\in\mathcal{B}^+}((E_n^r)^2 + E_n^2(t) + 2g_n(t))\} \\ &\leq B + \sum_{n\in\mathcal{B}^+}\mathbb{E}\{g_n(t)\}\end{aligned} \tag{21}$$

where $B = \frac{1}{2}\mathbb{E}\{\sum_{n\in\mathcal{B}^+}((E_n^r)^2 + E_n^2(t))\}$ is a constant and $g_n(t) = H_n(t)\mathbb{E}\{(E_n(t) - E_n^r)\}$.

Therefore, we convert the original problem (15) to minimize the upper-bound which not need future information:

$$\begin{aligned}&\min_{\boldsymbol{x},\boldsymbol{f}} \psi(x,f) = \mathbb{E}\{V\sum_{n\in\mathcal{B}} T_n(t) + \sum_{n\in\mathcal{B}^+} g_n(t)\} \\ &s.t.\ (15b),(15c),(15d),(15e)\end{aligned} \tag{22}$$

So far, we have overcome the challenge of requiring future information. However, problem (22) is still knotty to get an optimal solution because of non-convex objective and non-linear constraint. Besides, it is a problem with mixture of discrete variable and continuous variable. Thus, we further transform it, as shown in **Lemma 2**.

Lemma 2. *The optimization problem (22) is equivalent to the following problem:*

$$\begin{aligned}&\min_{\boldsymbol{x},\boldsymbol{f}} \phi(x,f) = p(x,f) + \sum_{n\in\mathcal{B}^+}\sum_{k\in\mathcal{K}} H_n(t)E_{n,k}^M(t)x_{n,k}(t) \\ &s.t.\ (15b),(15c),(15e)\end{aligned} \tag{23}$$

where $p(x,f) = \sum_{n\in\mathcal{B}^+}\sum_{k\in\mathcal{K}} H_n(t)(E_{n,k}^S(t) + E_{n,k}^C(t) - E_n^r) + \sum_{n\in\mathcal{B}} VT_n(t)$.

The proof of **Lemma 2** is shown in Appendix B. □

We further solve it through iterative optimization, which is described in detail in the next section.

5. PRACTICAL SOLUTION

This section introduces the MA-ECC algorithm in order to solve the multicast-aware caching and computing problem. MA-ECC determines which services are cached locally and how to allocate computing resources. First the problem is decomposed (23) into two sub-problems: optimization of computing resource allocation for a particular caching status and optimization of storage space allocation. MA-ECC is first described in the context of the two sub-problems, then its performance is analyzed theoretically.

5.1. Optimization of Caching Decision

This sub-problem concerns the allocation of storage space. The multicast-aware caching and computing problem takes the following form for a given computing resource allocation $\boldsymbol{f(t)}$.

$$\begin{aligned}\min_{\boldsymbol{x}} &\sum_{k\in\mathcal{K}}\{\sum_{n\in\mathcal{B}} V(T_{n,k}^M(t) + T_{n,k}^C(t))S_{n,k}(t) + \\ &\sum_{n\in\mathcal{B}^+} H_n(t)(E_{n,k}^M(t)x_{n,k}(t) + E_{n,k}^C(t) + E_{n,k}^S(t))\} \\ &s.t.(15b),(15e)\end{aligned} \tag{24}$$

The objective and constraints of problem (24) are linear which means this is a zero-one Type Integer Linear programming. By employing the *implicit enumeration method*, we can get the optimal caching decision of SBSs. This is briefly described in **Algorithm 1**.

5.2. Optimization of Computing Resource Allocation

The computing resource allocation is concerned in this subsection. The optimization problem for a given caching decision $\boldsymbol{x}$ is as follows:

$$\begin{aligned}&\min_{\boldsymbol{f}} \sum_{n\in\mathcal{B}^+}\sum_{k\in\mathcal{K}}(H_n(t)E_{n,k}^C(t) + VT_{n,k}^C(t)) \\ &s.t.\ (15c)\end{aligned} \tag{25}$$

Lemma 3. *Problem (25) is a convex problem.*

Algorithm 1: Optimization of Caching Decision

Input:
Nonnegative penalty parameter V;
Virtual queue length $H_n(t)$;
The requests $R_{n,k}(t)$;
The computing resource allocation $\boldsymbol{f(t)}$;
Output:
Caching status $\boldsymbol{x(t)}$;
Calculate the coefficients of (24) according to $\boldsymbol{f(t)}$;
Replace $\boldsymbol{x(t)}$ with $\boldsymbol{y(t)}$ to make all coefficients not less than 0;
Arrange the coefficients in ascending order;
Get the solution $\boldsymbol{y(t)}$ by the implicitimplicit enumeration method;
Change $\boldsymbol{y(t)}$ to $\boldsymbol{x(t)}$;
Return $\boldsymbol{x(t)}$;
final ;

The proof of **Lemma 3** is shown as Appendix C. □

As it is a convex problem, *Karush-Kuhn-Tucher (KKT)* can solve it. We define Lagrangian functions under inequality constraints:

$$L(\boldsymbol{f(t)}, \boldsymbol{\mu}) = \sum_{n\in\mathcal{B}^+}\sum_{k\in\mathcal{K}}\{H_n(t)E^C_{n,k}(t) + VT^C_{n,k}(t)\}+ \sum_{n\in\mathcal{B}^+}\{\mu_n(\sum_{k\in\mathcal{K}} f_{n,k}(t) - F_n)\} \tag{26}$$

The optimal solution $\boldsymbol{f^*(t)}$ should satisfy the following conditions:

$$\begin{cases} \frac{\partial L(\boldsymbol{f(t)},\boldsymbol{\mu})}{\partial \boldsymbol{f(t)}}|_{\boldsymbol{f(t)}=\boldsymbol{f^*(t)}} = 0 \\ \mu_n(\sum\limits_{k\in\mathcal{K}} f^*_{n,k}(t) - F_n) = 0 \\ \sum\limits_{k\in\mathcal{K}} f^*_{n,k}(t) - F_n \le 0 \\ \mu_n \ge 0 \end{cases} \tag{27}$$

By solving the equations, we get the optimal computing resource allocation $\boldsymbol{f^*(t)}$ for a given $\boldsymbol{x(t)}$.

5.3. Multicast-Aware Caching and Computing

The optimization solution of the multicast-aware caching and computing resource allocation problem (23) to minimize both the service latency and energy consumption is presented in detail by **Algorithm 2**. In each time slot t, we first use the average computing resource to initialize each service. We solve two sub-problems (24) and (25) because a much faster convergence can be achieved by using the solutions of the two sub-problems. Then, alternating iteration continues until convergence. We will get the solution for a given error tolerance ϵ. Finally, the energy queues are updated to prepare for the next time slot.

5.4. Performance Analysis

This subsection analyzes theoretically the performance of MA-ECC as shown in **Lemma 4**.

Lemma 4. *By applying MA-ECC, we have following performance guarantees:*

Algorithm 2: Multicast-Aware Edge Caching and Computing Algorithm (MA-ECC)

Input:
Nonnegative penalty parameter V;
Initialize queue length: $H_n(0) = 0, \forall n$;
Storage space $\boldsymbol{C}$, maximum frequency $\boldsymbol{F}$;
The storage and multicast size of service k: c_k and m_k;
The requests: $R_{n,k}(t)$;
The computing resource allocation of MBS: $f_{0,k}(t)$;
Number of iterations $w = 0$, maximum tolerance $\epsilon > 0$;
Output:
Caching status $\boldsymbol{x(t)}$ and computing resource $\boldsymbol{f(t)}$;
while $t \in \mathcal{T}$ **do**
  Initialize computing resource $f_{n,k}(t) = \frac{F_n}{|\mathcal{K}|}$
  while $|\frac{\boldsymbol{\phi_w(x(t),f(t))}-\boldsymbol{\phi_{w-1}(x(t),f(t))}}{\boldsymbol{\phi_{w-1}(x(t),f(t))}}| \ge \epsilon$ **do**
    Based on $\boldsymbol{f(t)}$, get the caching status $\boldsymbol{x(t)}$ by calling **Algorithm 1**;
    Update the computing resource allocation $\boldsymbol{f(t)}$ for given $\boldsymbol{x(t)}$ using KKT;
    Update $w = w + 1$;
  end
  $H_n(t+1) = max[H(t) - E^r_n + E_n(t), 0]$;
end
final ;

1) The difference between MA-ECC and the optimal algorithm in service latency satisfies:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{T(t)\} \le T^{opt} + \frac{B+C}{V} \tag{28}$$

where C is a positive constant, $T(t) = \sum\limits_{n\in\mathcal{B}} T_n(t)$ is the sum of latency of all BS and T^{opt} is the infimum average latency time achievable by any policy that satisfies constraints.

2) The energy queue length which is the time-average deviation of energy consumption satisfies:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{n\in\mathcal{B}}\mathbb{E}\{H_n(t)\} \le \frac{B+C+V[\Phi(\varepsilon)-T^{opt}]}{\varepsilon} \tag{29}$$

where ε and $\Phi(\varepsilon)$ are constants that satisfy the Slater condition [28]. Besides, $\varepsilon > 0$ and $T^{min} \le \Phi(\varepsilon) \le T^{max}$, where T^{min} and T^{max} are finite constants and minimum and maximum bounds for $T(t)$.

Lemma 4 demonstrates that the services latency achieved by the MA-ECC algorithm diverges from the optimal solution with $O(1/V)$. However, the energy queue length which represents the energy deficit is bound by $O(V)$. There is a tradeoff between services latency and energy consumption by parameter V. When parameter V becomes large, the service latency is more emphasized and MA-ECC achieved a better performance with consuming more energy.

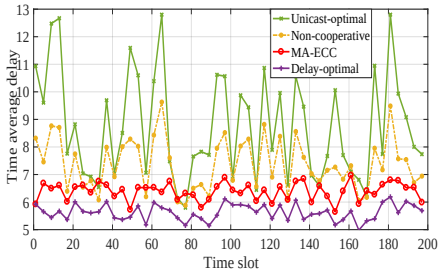

Figure 3: Time average delay

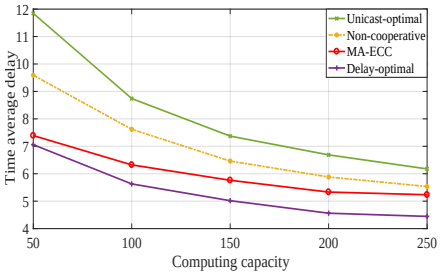

Figure 4: Time average delay with F_n

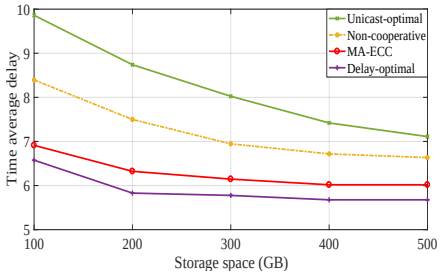

Figure 5: Time average delay with C_n

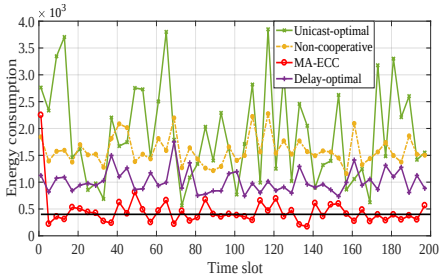

Figure 6: Energy consumption

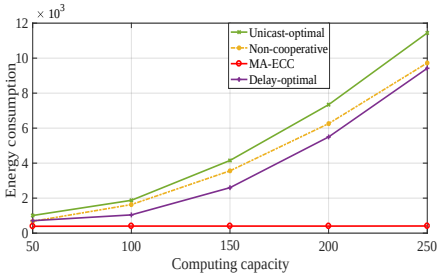

Figure 7: Energy consumption with F_n

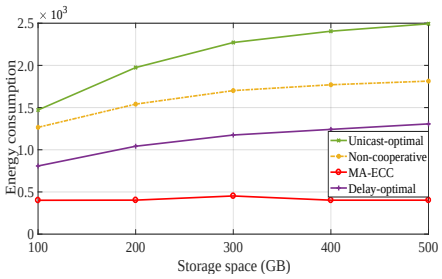

Figure 8: Energy consumption with C_n

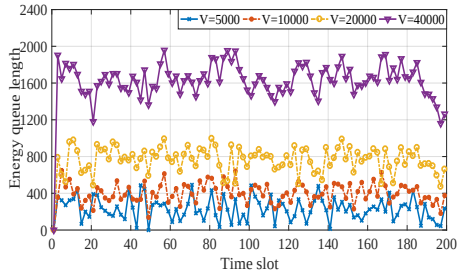

Figure 9: Length of all energy queue

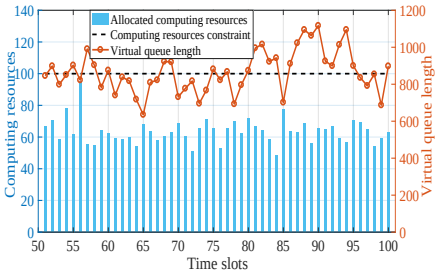

Figure 10: Queue length and workload

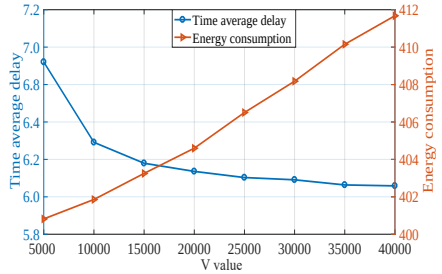

Figure 11: Effect of V value

6. SIMULATION RESULTS

Extensive simulations were designed and run to demonstrate the performance of MA-ECC. We consider a $1000m \times 1000m$ area consisting of one MBS and 7 SBSs. We fix the MBS at the center of the network and randomly distribute the SBSs. There are $K = 20$ independent services and the request frequency conforms to the Zipf distribution with the parameter of 0.95 [29]. Besides, we formulate the total service request in BS b_n as a Poisson process. Other main parameter settings are shown in Table 2.

Table 2: Parameter settings for simulations

Parameters	Value
Communicate bandwidth of MBS	2 GB
Communicate bandwidth of SBS	6 GB
BS storage space C_n	200 GB
Storage space for service k, c_k	[2,40] GB
Unit energy consumption c_k	0.04 kWh
SBS computing resource F_k	100 GHz
Computing resource requirement of service k, d_k	[5,15] GHz

In order to evaluate performance, we compare MA-ECC with the following solutions.

- **Unicast-optimal resource allocation**: To demonstrate the advantages brought by considering multicast in service latency and energy consumption, we remove constraints on energy consumption in algorithm [21], making it the optimal algorithm in unicast.
- **Non-cooperative resource allocation**: The most popular services in the serving region are stored in SBS. There are not mutual communications between SBSs and we ignore the constraint of energy consumption.
- **Delay-optimal resource allocation**: Removing the limited energy consumption, we find a global optimal resource allocation decision for all the BSs and time slot to minimize the service delay.

6.1. Average delay comparison

Fig.3-Fig.5 show the time average delay in different conditions. Specifically, Fig.3 shows the dynamic change of time delay at different time slots. It shows that MA-ECC achieves time average delay approximate to the delay-optimal resource allocation and significantly better than non-cooperative resource allocation and unicast-optimal resource allocation. For the no-cooperative resource allocation, BSs individually cache the

most demanding services in the serving region, which lacks the future information and the cooperation between BSs. The method falls into a local optimal solution and results in poor performance in time average delay. In the unicast-optimal cast, BSs have to transmit the service to the user one by one, which leads to higher communication time. Therefore, it has the worst performance in terms of time average delay.

Fig.4 shows the impact of computing capacity on time average delay. We see that the system delay decreases with the increase of SBS computing capacity for all four methods. Obviously, the reason is that as the computing capacity increases, the computation time decreases which influences the time average delay. However, because of the long-term energy consumption constraint, the downward trend of MA-ECC is more gradual than the other three methods, which means when the computing capacity is sufficient, the energy consumption will become the bottleneck for MA-ECC to reduce the system delay.

Fig.5 presents time average delay variation with different storage space. Similar to computing capacity, increasing the storage space of SBSs can reduce the time average delay since more services can be cached in SBSs. However, the effect will gradually weaken, because storage space can affect communication time, but has less impact on computing time due to the limited computing resources.

6.2. Energy consumption comparison

Fig.6 shows the energy consumption of the four algorithms at different time slots. We observe that the energy consumption of MA-ECC fluctuates around the limited power E_n^r (the black line), while the other three algorithms are far beyond the constraint. As expected, although the delay-optimal algorithm has the lowest time average delay, it consumes two times more energy than MA-ECC. Non-cooperative resource allocation runs out of storage space by caching the most popular service at each time slot, which leads to higher caching energy consumption. Compared with multicast schemes, unicast-optimal algorithm needs more energy to support the transmission of multiple data streams.

Fig.7 and Fig.8 present the impact of computing capacity and storage space on system energy consumption, respectively. In Fig.7, we find the energy consumption of MA-ECC closely follows the constraint in any case, but the other three algorithms overuse energy. Besides, the trend of the other three graphs is close to the square growth because of equation (11). As for storage space, Fig.8 shows that the energy consumption increases as the storage space since more services can be cached and provided by SBSs. However, the rate of increase will gradually decrease until reaches zero.

6.3. Impact of V value

Fig.9 illustrates the impact of V on energy queues length which indicates the backlog of deviation of current energy consumption from the energy constraint. We find that the length of energy queues becomes longer as V increases. In other words, the energy deficit increases with V. The reason is that the time average delay becomes more important in the optimization problem (23) with increasing V and MA-ECC will achieve

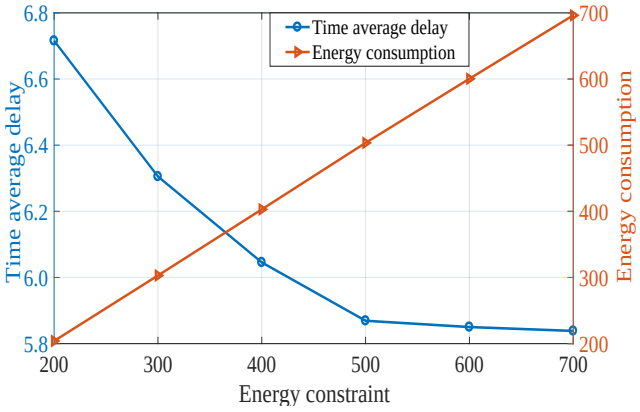

Figure 12: Effect of energy constraint

a lower system delay at a cost of higher energy deficit. We also observe that the energy queue length is proportional to V, which is consistent with the second part of **Lemma 4**. In order to show more intuitively the impact of energy queue length on network resource allocation, we present the energy queue for $V = 20000$ and computing resource usage in Fig.10 at the same time. As the figure shows, SBSs tend to use more computing resources to minimize time average delay when the energy queue length is small (e.g. slot 65, 85). On the contrary, if the energy length is large, SBSs allocate lower computing resources to reduce the energy consumption (e.g. slot 57, 84). Therefore, the energy queue can affect resource allocation to make a tradeoff between service latency and energy consumption.

Fig.11 shows the impact of V on time average delay and energy consumption. It shows that the system delay is inversely proportional to V, which proves the first part of **Lemma 4** experimentally. In contrast, energy consumption is roughly proportional to V. This can be explained by equation (16) and the second part of **Lemma 4**.

6.4. Impact of energy constraint

Fig.12 illustrates the impact of energy consumption constraints E_n^r on the performance of MA-ECC. As the picture shows, the MA-ECC reduces the time average delay with the increase of energy consumption constraint, because SBSs can allocate more resources to satisfy users' requests. However, due to the limited resource, the performance gain becomes more gentle when the constraint E_n^r is large. Besides, we also observe from the right-side of y-axis that MA-ECC successfully meets the predetermined energy consumption constraint.

6.5. Convergence of algorithm

In Fig.13, the convergence of the proposed MA-ECC is evaluated under different initial parameters. We set error tolerance parameter ϵ as 10^{-2}. In order to simplify, we unified the initial parameters of each SBS. Specially, E_r is the limited power of SBSs, C is the storage space of SBSs and F is the computing capacity. The x axis is the iteration number of MA-ECC algorithm and y axis is the objective function value of the optimization problem (23). The pictures shows that although the

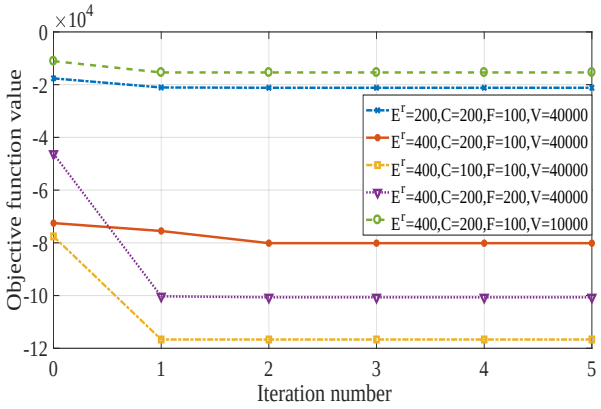

Figure 13: Convergence of the MA-ECC

initial parameters are different, MA-ECC usually achieves convergence within 3 iterations, which means that MA-ECC is efficient in terms of computational complexity.

7. Conclusion

In this paper, we studied multicast-aware resource allocation with joint optimization of caching and computing for MEC in a heterogeneous cellular network environment. First, we formulate it as a convex optimization problem. Then, in order to effectively solve the problem, we separate it into two subproblems, optimization of caching decision and optimization of computing resource allocation. Additionally, we propose an online algorithm MA-ECC as a solution to the overall problem. Theoretical analysis shows our solution not only reduces effectively the time average delay, but also keeps low the energy consumption. The experimental simulation-based performance evaluation shows our proposed solution outperforms three alternative solutions under different system parameters.

Future work will study the computation offloading on the basis of the original problem and study the multicast-aware computation offloading and resource allocation for MEC.

Acknowledgment

This work is supported by National Key R&D Program of China (2018YFE0205502); the National Natural Science Foundation of China (NSFC) under Grant Nos. 61871048 and 61872253; the BUPT Excellent Ph.D. Students Foundation CX2018108.

References

[1] J. S. Pan, A trust game model of service cooperation in cloud computing, Journal of Network and Computer Applications 173 (2021).

[2] L. Hai, Z. Sherali, C. Zhihong, L. Houda, W. Lusheng, A survey on computation offloading modeling for edge computing, Journal of Network and Computer Applications 169 (2020).

[3] C. Wang, C. Liang, F. R. Yu, Q. Chen, L. Tang, Computation offloading and resource allocation in wireless cellular networks with mobile edge computing, IEEE Transactions on Wireless Communications 16 (8) (2017) 4924–4938.

[4] S. JoÅilo, G. Dn, Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks, IEEE Transactions on Mobile Computing 18 (1) (2019) 207–220.

[5] U. Saleem, Y. Liu, S. Jangsher, X. Tao, Y. Li, Latency minimization for d2d-enabled partial computation offloading in mobile edge computing, IEEE Transactions on Vehicular Technology 69 (4) (2020) 4472–4486.

[6] X. Xiaolong, L. Yuancheng, H. Tao, X. Yuan, P. Kai, Q. Lianyong, D. Wanchun, An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks, Journal of Network and Computer Applications 133 (2019) 75–85.

[7] S. Guo, L. Jiadi, Y. Yang, B. Xiao, Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing, IEEE Transactions on Mobile Computing 18 (2) (2019) 319–333.

[8] Y. Wei, F. R. Yu, M. Song, Z. Han, Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actorcritic deep reinforcement learning, IEEE Internet of Things Journal 6 (2) (2019) 2061–2073.

[9] K. Shashwat, D. Sai Vineeth, F. A. Antony, J. Jiong, Ran-aware adaptive video caching in multi-access edge computing networks, Journal of Network and Computer Applications 168.

[10] R. W. L. Coutinho, A. Boukerche, Modeling and analysis of a shared edge caching system for connected cars and industrial iot-based applications, IEEE Transactions on Industrial Informatics 16 (3) (2020) 2003–2012.

[11] D. Wu, Q. Liu, H. Wang, Q. Yang, R. Wang, Cache less for more: Exploiting cooperative video caching and delivery in d2d communications, IEEE Transactions on Multimedia 21 (7) (2019) 1788–1798.

[12] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, X. Shen, Statistically indifferent quality variation: An approach for reducing multimedia distribution cost for adaptive video streaming services, IEEE Transactions on Multimedia 21 (4) (2019) 915–929.

[13] C. Li, M. Song, S. Du, X. Wang, Z. Min, Y. Luo, Adaptive priority-based cache replacement and prediction-based cache prefetching in edge computing environmen, Journal of Network and Computer Applications 165 (2020).

[14] G. Araniti, F. Rinaldi, P. Scopelliti, A. Molinaro, A. Iera, A dynamic mbsfn area formation algorithm for multicast service delivery in 5g nr networks, IEEE Transactions on Wireless Communications 19 (2) (2020) 808–821.

[15] Z. Jiang, C. Xu, J. Guan, Y. Liu, G.-M. Muntean, Stochastic analysis of dash-based video service in high-speed railway networks, IEEE transactions on Multimedia 21 (6) (2019) 1577–1592.

[16] Y. Lu, W. Chen, H. V. Poor, Coded joint pushing and caching with asynchronous user requests, IEEE Journal on Selected Areas in Communications 36 (8) (2018) 1643–1856.

[17] C. Desogus, M. Anedda, M. Fadda, M. Murroni, Additive logarithmic weighting for balancing video delivery over heterogeneous networks, IEEE Transactions on Broadcasting.

[18] L. Wei, F. C. Heng, B. He, J. Cai, Towards efficient resource allocation for heterogeneous workloads in iaas clouds, IEEE Transactions on Cloud Computing 6 (1) (2018) 264–275.

[19] C. Liang, Y. He, F. R. Yu, N. Zhao, Enhancing video rate adaptation with mobile edge computing and caching in software-defined mobile networks, IEEE Transactions on Wireless Communications 17 (2017) 76–88.

[20] L. Yang, J. Cao, G. Liang, X. Han, Cost aware service placement and load dispatching in mobile cloud systems, IEEE Transactions on Computers 65 (5) (2016) 1440–1452.

[21] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Chen, B. Hu, V. C. M. Leung, Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching, IEEE Internet of Things Journal 6 (3) (2019) 4283–4294.

[22] B. Zhou, Y. Cui, M. Tao, Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks, IEEE Transactions on Communications 65 (7) (2017) 2956–2970.

[23] C. Xu, Z. Li, L. Zhong, H. Zhang, G.-M. Muntean, Cmt-nc: Improving the concurrent multipath transfer performance using network coding in wireless networks, IEEE Transactions on Vehicular Technology 65 (3) (2016) 1735–1751.

[24] C. Wang, F. R. Yu, C. Liang, Q. Chen, L. Tang, Joint computation offloading and interference management in wireless cellular networks with mobile edge computing, IEEE Transactions on Vehicular Technology 66 (8) (2017) 7432–7445.

[25] S. Li, N. Zhang, S. Lin, L. Kong, A. Katangur, K. M. Khurram, M. Ni, G. Zhu, Joint admission control and resource allocation in edge computing for internet of things, IEEE Network 32 (1) (2018) 72–79.
[26] Z. Tan, F. R. Yu, X. Li, H. Ji, V. C. M. Leung, Virtual resource allocation for heterogeneous services in full duplex-enabled scns with mobile edge computing and caching, IEEE Transactions on Vehicular Technology 67 (2) (2018) 1794–1808.
[27] Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing: Partial computation offloading using dynamic voltage scaling, IEEE Transactions on Communications 64 (10) (2016) 4268–4282.
[28] M. J. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems, Morgan & Claypool, 2010.
[29] M. Zhang, H. Luo, H. Zhang, A survey of caching mechanisms in information-centric networking, IEEE Communications Surveys & Tutorials 17 (3) (2015) 1473–1499.

AppendixA. PROOF OF LEMMA 1

For base station b_n, we have the inequality $H_n(t+1)-H_n(t) \geq E_n(t) - E_n^r$ according to equation (16). Then we obtain the equation (A.1) by the summation of this inequality over $t \in \{0, ...T-1\}$:

$$\frac{H_n(T)}{T} - \frac{H_n(0)}{T} \geq \frac{1}{T}\sum_{t=0}^{T-1} E_n(t) - E_n^r \tag{A.1}$$

Considering $H_n(0) = 0$, we have the following inequality by taking expectation of equation (A.1) and taking $T \to \infty$:

$$\lim_{T\to\infty} \frac{\mathbb{E}\{H_n(T)\}}{T} \geq \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E_n(t) - E_n^r \tag{A.2}$$

The energy queue $H_n(t)$ is mean rate stable which means $\lim_{t\to\infty} \frac{\mathbb{E}\{H_n(t)\}}{t} = 0$. Combined with equation (A.1), the constraint $\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E_n(t) \leq E_n^r$ is satisfied. Therefore, **Lemma 1** is proved. □

AppendixB. PROOF OF LEMMA 2

The $\psi(x, f)$ is composed of service latency and energy consumption. According to equation (14), we have

$$\psi(x, f) = p(x, f) + \sum_{n\in\mathcal{B}^+}\sum_{k\in\mathcal{K}} H_n(t)E_{n,k}^M(t)\mathbf{1}_{\{f_{n,k}(t)\neq 0\}} \tag{B.1}$$

Compare with $\phi(x, f)$, we find that if the optimal solution $(\boldsymbol{x^*}, \boldsymbol{f^*})$ of problem (22) satisfies $\mathbf{1}_{\{f_{n,k}^*(t)\neq 0\}} = x_{n,k}^*(t)$, then constraint (15d) is useless and problem (22) is equivalent to problem (23). Now, we first remove constraint (15d) and discuss the problem separately.

Case one: $\mathbf{1}_{\{f_{n,k}^*(t)\neq 0\}} > x_{n,k}^*(t)$, which means $f_{n,k}^*(t) > 0$ and $x_{n,k}^*(t) = 0$. Then we construct another strategy $(x_{n,k}^*(t), f_{n,k}^+(t) = 0)$ which also satisfies the constraints. According to equation (1), the requests for service k to b_n is offloaded to MBS, which means b_n can provide service k to nobody. By calculating, we find $p(x^*, f^*) = p(x^*, f^+)$. As for multicast, the energy consumption of $f_{n,k}^*(t)$ are larger than $f_{n,k}^+(t)$. In other words, we have $\psi(x^*, f^*) > \psi(x^*, f^+)$ (strategy (x^*, f^+) is better than (x^*, f^*)), which is conflicts with the assumption that $(\boldsymbol{x^*}, \boldsymbol{f^*})$ is optimal. Therefore, case one in not true.

Case two: $\mathbf{1}_{\{f_{n,k}^*(t)\neq 0\}} < x_{n,k}^*(t)$, which means $f_{n,k}^*(t) = 0$ and $x_{n,k}^*(t) = 1$. First, we start the case where the request number $R_{n,k}(t) \neq 0$. In this case, the actual service volume $S_{n,k}(t) \neq 0$ due to $x_{n,k}^*(t) = 1$. But the computation time is infinity because of $f_{n,k}^*(t) = 0$. This means $\psi(x^*, f^*) \to \infty$ which is clearly conflicts with the assumption. If the request number $R_{n,k} = 0$, we construct another strategy $(x_{n,k}^+(t) = 0, f_{n,k}^*(t))$ which also satisfies the constraints. There are no requests for service k, so the costs of computation and multicast are zero for the two policy. However, the caching energy consumption of $(f_{n,k}^*(t), x_{n,k}^+(t))$ is smaller than $(f_{n,k}^*(t), x_{n,k}^*(t))$ which need to cache service k. Therefore, the cost function $\psi(x^*, f^*) > \psi(x^+, f^*)$, which is also conflicts with the assumption. Therefore, case two is false.

In conclusion, the optimal solution $(\boldsymbol{x^*}, \boldsymbol{f^*})$ satisfies $\mathbf{1}_{\{f^*_{n,k}(t)\neq 0\}} = x^*_{n,k}(t)$ which means the constraint (15d) is useless and the objective function $\psi(x, f) = \phi(x, f)$. Therefore, the problem (22) is equivalent to problem (23) and **Lemma 2** is proved. □

AppendixC. PROOF OF LEMMA 3

For $\forall f_i \in \boldsymbol{f}$, we have the second partial derivative of $p(\boldsymbol{f})$ with respect to f_i:

$$\frac{\partial^2 p}{\partial f_i^2} = 2d_k S_{n,k}(t)\zeta_n + \frac{2d_k S_{n,k}(t)}{f_i^3} \geq 0 \tag{C.1}$$

For $\forall f_i, f_j \in \boldsymbol{f}$ and $i \neq j$, we have the second partial derivative:

$$\frac{\partial^2 p}{\partial f_i \partial f_j} = 0 \tag{C.2}$$

Therefore, we get the Hessian matrix as following:

$$h(p) = \begin{bmatrix} \frac{\partial^2 p}{\partial f_1^2} & 0 & \dots & 0 \\ 0 & \frac{\partial^2 p}{\partial f_2^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial^2 p}{\partial f_{NK}^2} \end{bmatrix} \tag{C.3}$$

The value on the diagonal of this matrix is non-negative and other elements of this matrix is zero, so the Hessian matrix is positive semidefinite which means the objective function is convex. Moreover, the constraints of problem (25) are linear. Therefore, problem (25) is a convex problem and **Lemma 3** is proved. □

AppendixD. PROOF OF LEMMA 4

According to the Theorem 4.5 in [28], there is a policy $(\boldsymbol{x}^\delta(t), \boldsymbol{f}^\delta(t))$ satisfies following inequalities for any $\delta > 0$:

$$\begin{aligned} \mathbb{E}\{T^\delta(t)\} &\leq T^{opt} + \delta \\ \mathbb{E}\{E_n^\delta(t) - E_n^r\} &\leq \delta \end{aligned} \tag{D.1}$$

Given equation (20) and (21), we have the following inequalities for policy $(\boldsymbol{x}^\delta(t), \boldsymbol{f}^\delta(t))$ based on the Definition 4.7 in [28]:

$$\begin{aligned} &\Delta q(t) + V\mathbb{E}\{T(t)\} \\ \leq &\Delta q(t) + V\mathbb{E}\{T^\delta(t)\} + C \\ \leq &B + \sum_{n\in\mathcal{B}} \mathbb{E}\{H_n(t)(E_n^\delta(t) - E_n^r)\} + V\mathbb{E}\{T^\delta(t)\} + C \\ \leq &B + \delta\sum_{n\in\mathcal{B}} \mathbb{E}\{H_n(t)\} + V(T^{opt} + \delta) + C \end{aligned} \tag{D.2}$$

Taking $\delta \to 0$:

$$\mathbb{E}\{T(t)\} \leq T^{opt} + \frac{B + C}{V} - \frac{\Delta q(t)}{V} \tag{D.3}$$

Considering $q(t) \geq 0$ and $q(0) = 0$, we sum above equations over all time slots, divide by T and take $T \to \infty$:

$$\begin{aligned} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{T(t)\} &\leq T^{opt} + \frac{B + C}{V} - \frac{q(t) - q(0)}{VT} \\ &\leq T^{opt} + \frac{B + C}{V} \end{aligned} \tag{D.4}$$

So far, we prove the first part of **Lemma 4.**

There is a policy $(\boldsymbol{x}^\varepsilon(t), \boldsymbol{f}^\varepsilon(t))$ satisfies the following Slater Condition:

$$\begin{aligned} \mathbb{E}\{T^\varepsilon(t)\} &= \Phi(\varepsilon) \\ \mathbb{E}\{E_n^\varepsilon(t) - E_n^r\} &\leq -\varepsilon \end{aligned} \tag{D.5}$$

Combining with formula (D.4), we have:

$$\begin{aligned} &\Delta q(t) + V\mathbb{E}\{T(t)\} \\ \leq &B + \sum_{n\in\mathcal{B}} \mathbb{E}\{H_n(t)(E_n^\varepsilon(t) - E_n^r)\} + V\mathbb{E}\{T^\varepsilon(t)\} + C \\ \leq &B - \varepsilon\sum_{n\in\mathcal{B}} \mathbb{E}\{H_n(t)\} + V\Phi(\varepsilon) + C \end{aligned} \tag{D.6}$$

Rearranging terms:

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n\in\mathcal{B}} \mathbb{E}\{H_n(t)\} \leq &\frac{1}{T\varepsilon}\{\mathbb{E}\{q(0) - q(t)\}\} + \\ &\frac{1}{\varepsilon}\{B + V[\Phi(\varepsilon) - \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{T_n(t)\}] + C\} \end{aligned} \tag{D.7}$$

The expectation of $T_n(t)$ cannot be less than T^{opt}. We take a limit as $T \to \infty$:

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n\in\mathcal{B}} \mathbb{E}\{H_n(t)\} \leq \frac{B + C + V[\Phi(\varepsilon) - T^{opt}]}{\varepsilon} \tag{D.8}$$

Therefore, the second part of **Lemma 4** is proved. □