sensors

MDPI

Article

AoI-Aware Optimization of Service Caching-Assisted Offloading and Resource Allocation in Edge Cellular Networks

Jialiang Feng and Jie Gong *

The Guangdong Key Laboratory of Information Security Technology, School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510006, China
* Correspondence: gongj26@mail.sysu.edu.cn

Abstract: The rapid development of the Internet of Things (IoT) has led to computational offloading at the edge; this is a promising paradigm for achieving intelligence everywhere. As offloading can lead to more traffic in cellular networks, cache technology is used to alleviate the channel burden. For example, a deep neural network (DNN)-based inference task requires a computation service that involves running libraries and parameters. Thus, caching the service package is necessary for repeatedly running DNN-based inference tasks. On the other hand, as the DNN parameters are usually trained in distribution, IoT devices need to fetch up-to-date parameters for inference task execution. In this work, we consider the joint optimization of computation offloading, service caching, and the AoI metric. We formulate a problem to minimize the weighted sum of the average completion delay, energy consumption, and allocated bandwidth. Then, we propose the AoI-aware service caching-assisted offloading framework (ASCO) to solve it, which consists of the method of Lagrange multipliers with the KKT condition-based offloading module (LMKO), the Lyapunov optimization-based learning and update control module (LLUC), and the Kuhn–Munkres (KM) algorithm-based channel-division fetching module (KCDF). The simulation results demonstrate that our ASCO framework achieves superior performance in regard to time overhead, energy consumption, and allocated bandwidth. It is verified that our ASCO framework not only benefits the individual task but also the global bandwidth allocation.

Keywords: edge computing; computation offloading; service caching; age of information; resource allocation

Citation: Feng, J. ; Gong, J. AoI-Aware Optimization of Service Caching-Assisted Offloading and Resource Allocation in Edge Cellular Networks. *Sensors* **2023**, *23*, 3306. https://doi.org/10.3390/s23063306

Academic Editors: Hao Wang, Małgorzata Kujawska, Vijayakumar Varadarajan, Han-Chieh Chao, Lidia Dobrescu, Sheng-Lyang Jang, Yi Wu, Wencheng Lai, Adam W. Skorek and Rashmi Bhardwaj

Received: 8 February 2023
Revised: 28 February 2023
Accepted: 17 March 2023
Published: 21 March 2023

1. Introduction

In recent decades, the Internet of things (IoT) has experienced rapid development and become ubiquitous in our daily lives. IoT devices have proliferated and evolved with advanced hardware architectures, and are being leveraged to create seamless networks that cover every corner of our globe [1]. Along with the development of IoT devices, a promising computing paradigm known as edge computing has arisen; this involves moving the location of computation from the central network to the network edge [2]. Moving the task execution from the cloud server to the multi-access edge computing (MEC) server (e.g., base station, access point) significantly alleviates the congestion of the core network and releases the burden of the cloud. Tasks with real-time requirements, computation-intensive characteristics, and high energy consumption (e.g., deep neural network (DNN)-based automatic license plate recognition) appear. Mobile devices where tasks are generated are constrained in terms of energy and computational capabilities (e.g., smartphones and unmanned aerial vehicles). Therefore, it is necessary to offload tasks to nearby MEC servers for remote execution [3], which is also known as computation offloading [4].

However, the exponential growth in the volume of offloaded data has led to increased traffic burdens on cellular networks, causing channel congestion. Under unstable network conditions, such as extremely high transmission latency, the performance of computation

offloading can drastically decline. A caching policy [5] is proposed to tackle this issue by proactively storing the service in IoT devices, including MEC servers and mobile devices, to reduce the traffic of the cellular network. If an IoT device caches the service libraries and parameters, the task can be directly processed. Hence, the task processing time can dramatically reduce [6]. A DNN-based task is executed by a corresponding service package, consisting of reliable libraries and network parameters. Since the MEC server and mobile devices process distinct types of tasks, it is impractical to proactively cache all types of services due to storage limits. They only carry out caching whenever a task is required to be executed, and the caches are stored within a restricted time horizon.

Machine learning plays a significant role in the wireless network [7]. Considering a distributed machine learning scenario [8], the DNN is trained in a distributed manner. Then, the trained parameters of the DNN are assembled on an application server. The application server gathers all of the trained parameters and further trains a global DNN. Since the new data are generated from mobile devices, the trained parameters are updated ceaselessly and the global DNN is retrained based on the newly gathered parameters at the end of every global training round. Thus, the global DNN always reflects the up-to-date trained parameters. However, mobile devices may not fetch the latest parameters in every round. Hence, the cached DNN model may be outdated, which should be updated to keep the model fresh. To measure the freshness of the global service parameters at the MEC servers and the mobile devices, we introduce the concept of AoI [9], which is defined as the elapsed time since the generation of the latest received global service parameters response. The global service parameters are generated by training at the end of every global training round. When the MEC server or mobile device is required to execute inference tasks, it first checks whether fresh service parameters exist. If the service parameters are stale, the MEC server or mobile device needs to request the application server to fetch the up-to-date trained parameters for inference task execution.

1.1. Challenges

To realize distributed machine learning and service caching, the following challenges should be addressed:

1.1.1. Cost of the Task

On the one hand, the inference task completion time needs to be less than its corresponding maximal tolerance deadline. Thus, minimizing the inference task completion time is necessary for real-time requirements. On the other hand, the inference tasks are generated on energy-constrained mobile devices, which carefully make the offloading decisions to minimize energy consumption. Therefore, it is challenging to minimize the cost of the inference task consisting of time delay and energy consumed.

1.1.2. Bandwidth Consumption of the Application Server

If IoT devices fetch the latest service parameters from the application server, it utilizes the limited wireless bandwidth of cellular networks. Therefore, there is a trade-off between the fetching time and the total available bandwidth. If the application preferentially guarantees the fetching time, the remained bandwidth is not enough to serve other applications, and vice versa. Thus, the challenge of time and bandwidth trade-off needs to be addressed.

1.1.3. Matching between Wireless Channels and IoT Devices

In a condition of limited bandwidth, the matching between the wireless bandwidth and IoT devices is significant enough to minimize the fetching time since an IoT device may experience diverse channel fading and co-channel interference on different wireless channels. Hence, it is the third challenge to match between wireless channels and IoT devices to further minimize the service fetching time.

1.2. Related Work

1.2.1. Offloading with Cache

Some works make offloading decisions by considering the cache technology. In [10], an algorithm was devised by taking into account the multi-cast opportunity with cache in a multi-user scenario. A computing offloading and content caching model was proposed to reduce the time delay in the internet of vehicles in [11]. In [12], an optimal computing offloading and caching policy was designed to minimize the latency in a hybrid mobile system. In [13], an approximation collaborative computation offloading scheme and a game-theoretic collaborative computation offloading scheme were devised to achieve better offloading performance and scale well with the increasing computation task numbers. The above works do not consider the age of the cache, which may degrade the QoS.

1.2.2. Cache of Data

In terms of data caching, existing works focus on frequently reused data to improve performance. In [14], a deep supervised learning method was adopted to make real-time decisions in a dynamic vehicle network. An online caching placement and prediction-based data pre-fetch method were designed in [15] to address the uncertainty of future task parameters. In [16], a cache deployment strategy in a large-scale Wi-Fi system was adopted to maximize the caching benefit and achieve better caching performance. In [17], a joint power allocation–caching problem was formulated to maximize the downlink performance in the caching FiWi network. However, these works do not take into account the caching of the service, which is crucial in the DNN-based task.

1.2.3. Cache of Service

With respect to service caching, a few works consider caching services to enhance system efficiency. In [18], an online caching algorithm was proposed to minimize the overall computation delay. An extremely compelling (but much less studied) problem was studied in MEC-enabled dense cellular networks in [19]. In [20], an online service caching algorithm was devised to achieve the optimal worst-case competitive ratio under homogeneous task arrivals. In [21], a cache placement algorithm was adopted to minimize the data traffic forwarded to the remote cloud. The above-mentioned works only studied the cache and did not combine it with offloading.

1.2.4. Age of Information

In regard to the age of information, some works focused on minimizing the AoI of the optimized goal. In [9], the concept of AoI was first proposed, and general methods were derived to calculate the age metric, which can be applied to broad types of service systems. Dynamic cache content update scheduling algorithms were designed to minimize the average AoI of the dynamic content delivered to the users in [22]. In [23], a dueling deep R-network-based status updating algorithm was proposed by combining the dueling deep Q-network and R-learning to minimize the average cost. In [24], an algorithm aimed to obtain an optimal trade-off between age and latency was adopted for the freshness-aware buffer update in a mobile edge scenario. However, these works did not leverage the AoI metric to improve the offloading performance in an edge system.

1.3. Contribution

In this paper, we consider an AoI-aware service caching-assisted offloading scenario. Our objective is to minimize the weighted sum of the average completion delay, energy consumption, and allocated bandwidth. We decompose the original problem into three subproblems: minimizing the average time overhead cost and energy consumption of inference tasks, minimizing the required average bandwidth, and minimizing the fetching time of responding IoT devices. Furthermore, to solve the subproblems, we propose the AoI-aware service caching-assisted offloading framework (ASCO) to deal with them, which consists of three modules: the method of Lagrange multipliers with the KKT condition-

based offloading module (LMKO), the Lyapunov optimization-based learning and update control module (LLUC), and the Kuhn–Munkres (KM) algorithm-based channel-division fetching module (KCDF). Simulation results show that our ASCO framework achieves superior performance compared to other baseline combinations in terms of time overhead, energy consumption, and allocated bandwidth. The main contributions of the paper are summarized as

1. To minimize the average time overhead cost and energy consumption of inference tasks, we transform the problem into a Lagrangian dual problem. Then, we propose the LMKO module based on the method of Lagrange multipliers with Karush–Kuhn–Tucker (KKT) conditions to make an optimal offloading decision.
2. To minimize the required average bandwidth, we transform the problem into a Lyapunov plus penalty problem by minimizing the total required bandwidth while keeping the requesting data queue backlog stable. Further, we propose the LLUC module based on the Lyapunov optimization to derive an optimal dequeued rate.
3. To minimize the fetching time of IoT devices, we consider the problem of finding the perfect matching by maximizing the sum of the link weights in the equalling subgraph. Moreover, we propose the KCDF module based on the KM algorithm to obtain the optimal matching decision.

The novelty of the paper consists of three aspects. First we propose an AoI-aware service caching-assisted offloading scenario, which has not been considered in the literature. This scenario takes into account the service caching in distributed machine learning, including the service libraries and parameters. It is a popular technology and worthy to be investigated. We also consider the freshness of the service caching for the computation offloading. Existing works omit the AoI of the caching, especially the service caching, which degrades the offloading performance. We aim to minimize the costs from both the mobile device side and the global perspective. Then, we propose the novel ASCO framework, including three modules. The proposed algorithm outperforms the existing baselines.

The rest of the paper is organized as follows. We elaborate on the system module in Section 2. An analysis of the formulated problems is detailed in Section 3. Section 4 presents the proposed solution. The evaluation simulation is described in Section 5, followed by the conclusion in Section 6.

2. System Model

We consider an AoI-aware service caching asymmetric network consisting of heterogeneous mobile devices and MEC servers in Figure 1. A set of $|\mathcal{N}|$ mobile devices indexed by n is denoted as $\mathcal{N} = \{1, \cdots, |\mathcal{N}|\}$, e.g., smartphones and intelligent vehicles. A set of $|\mathcal{M}|$ MEC servers indexed by m is denoted as $\mathcal{M} = \{1, \cdots, |\mathcal{M}|\}$, e.g., access points and base stations. Since an AI-based inference task generated from the mobile device is computation-intensive and has real-time requirements, the mobile device with constrained computation capability needs to offload the inference task to the MEC server with sufficient computation resources. On the one hand, the inference task is processed by the corresponding service. For instance, an image recognition inference task is inferred by a DNN service running in service libraries, e.g., machine learning frameworks. On the other hand, caching data can alleviate the transmission traffic during the offloading and include content caching and service caching.

Considering a distributed machine learning scenario, an application server periodically trains an up-to-date DNN and then distributes it to the MEC servers and mobile devices, among which the inference task data are hardly reusable while the DNN service is frequently reusable. Thus, different from cashing the inference task data, caching the DNN service significantly reduces the transmission time. Note that the DNN service consists of the service libraries and service parameters. Since a DNN with the latest parameters owns better inference accuracy based on the periodical training, the service parameters should be updated when it has a new version. The service libraries are static and are only transmitted once for caching while the service parameters are dynamic. We define the AoI

as the elapsed time since the generation of the latest received service parameters at MEC servers or mobile devices, which measures the freshness of service parameters. If the AoI of service parameters is less than the periodical training round, then the parameters are considered to be the latest version and fresh enough to be used for inference. Otherwise, since a new version is generated at the application server, the parameters are stale and need to be updated to the latest version. Note that mobile devices do not hold the AoI information on the side of MEC servers due to privacy concerns and transmission overhead.

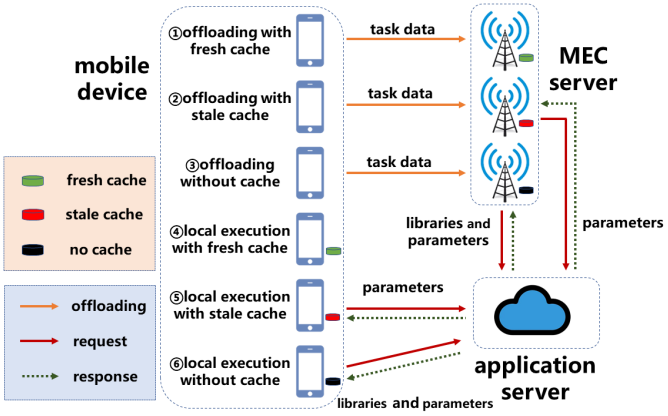


Figure 1. Schematics of the offloading cases.

2.1. Task Model

Considering a time-slotted system, a set of $|\mathcal{T}|$ timeslots indexed by t is denoted as $\mathcal{T} = \{1, \cdots, |\mathcal{T}|\}$. The inference task generated from the mobile device n at timeslot t is denoted as $k_n(t)$. A set of $|\mathcal{J}|$ service types indexed by j is denoted as $\mathcal{J} = \{1, \cdots, |\mathcal{J}|\}$. For instance, plate image recognition and face recognition are distinct types of services. Each inference task has a corresponding service type; the relationship is represented as follows: $x^{\text{typ}}_{k_n(t),j} = 1$ if $k_n(t)$ is of type j; otherwise, $x^{\text{typ}}_{k_n(t),j} = 0$. This satisfies the condition that each inference task can only be executed by a service of a certain type, at most: $\sum_{j \in \mathcal{J}} x^{\text{typ}}_{k_n(t),j} = 1$.

The inference tasks are computation-intensive and have maximum time tolerance. We define the inference task profile of $k_n(t)$ as $(d_{k_n(t)}, c_{k_n(t)}, T^{\max}_{k_n(t)})$, where $d_{k_n(t)}$ is the inference task input size, $c_{k_n(t)}$ is the task computation amount, and $T^{\max}_{k_n(t)}$ is the task maximum completion tolerance deadline. Take the task image recognition as an example, $d_{k_n(t)}$ is the image bit size, and $c_{k_n(t)}$ represents the required CPU cycles of the DNN service. $T^{\max}_{k_n(t)}$ is the image recognition deadline, meaning that the inference task processing delay cannot exceed the tolerance time.

2.2. Communication Model

The application server, mobile devices, and MEC servers mutually communicate under the cellular network. Based on the Shannon theory, the transmission rate between the mobile device n and MEC server m can be referred to as

$$r_{n,m}(t) = b_{n,m}(t) \log_2\left(1 + \frac{p_{n,m}(t) h_{n,m}(t)}{\sigma^2 + I_{n,m}(t)}\right), \tag{1}$$

where $b_{n,m}(t)$ is the allocation bandwidth, $p_{n,m}(t)$ means the transmission power from n to m, $h_{n,m}(t)$ is the channel gain, σ^2 represents the additive white Gaussian noise, and $I_{n,m}(t)$ is the co-channel interference that mobile device n suffers on the cellular channel, respectively. The transmission power affects the achievable spectral efficiency, and a highly allocated bandwidth can lead to an efficient transmission rate. The channel gain between each one

varies due to mobility. Since mobile devices are energy-constrained, the transmission power has an upper bound: $p_{n,m}(t) \leq p^{\max}$, where $p^{\max}$ is the maximum transmission power.

Moreover, the uploading time of inference task $k_n(t)$ from the mobile device n to the MEC server m can be calculated as

$$T^{\text{upl}}_{n,m,k_n(t)} = x^{\text{exe}}_{m,k_n(t)}(t)\frac{d_{k_n(t)}}{r_{n,m}(t)}, \tag{2}$$

where $x^{\text{exe}}_{m,k_n(t)}(t)$ is the offloading decision and defined as $x^{\text{exe}}_{m,k_n(t)}(t) = 1$ if $k_n(t)$ is offloaded to m; otherwise, $x^{\text{exe}}_{m,k_n(t)}(t) = 0$. It satisfies: $\sum_{m\in\mathcal{M}} x^{\text{exe}}_{m,k_n(t)}(t) \leq 1$, meaning that each inference task is offloaded to one MEC server at most.

The transmission rate from the application server to the MEC server $r_{0,m}(t)$ and the transmission rate from the application server to the mobile device $r_{0,n}(t)$ can be similarly calculated with (1).

2.3. Caching Model

For the purpose of alleviating the transmission traffic, the MEC servers and mobile devices have to cache the DNN service in their caching storage if they have no corresponding cache. Let d^{lib}_j be the library size and d^{par}_j be the parameter size of service type j, respectively.

Therefore, the fetching time for the DNN service of type j to the MEC server m can be calculated as

$$T^{\text{cac}}_{0,m,j} = T^{\text{lib}}_{0,m,j} + T^{\text{par}}_{0,m,j} = (1 - x^{\text{cac}}_{m,j}(t))(\frac{d^{\text{lib}}_j}{r_{0,m}(t)} + \frac{d^{\text{par}}_j}{r_{0,m}(t)}), \tag{3}$$

where $T^{\text{lib}}_{0,m,j}$ and $T^{\text{par}}_{0,m,j}$ are the fetching times of the service libraries and parameters at the MEC servers, respectively, and $x^{\text{cac}}_{m,j}(t)$ is the service caching placement decision at the MEC server m, defined as $x^{\text{cac}}_{m,j}(t) = 1$ if j is cached in m; otherwise, $x^{\text{cac}}_{m,j}(t) = 0$. Similarly, the fetching time for the DNN service of type j to mobile device n can be represented as $T^{\text{cac}}_{0,n,j}$.

Due to the limited caching capacity of the MEC server, there is a constraint on the storage cache:

$$\sum_{j\in\mathcal{J}} x^{\text{cac}}_{m,j}(t)(d^{\text{lib}}_j + d^{\text{par}}_j) \leq d^{\max}_m, \tag{4}$$

where the total DNN service size of all types cannot exceed the storage upper bound $d^{\max}_m$. The total DNN service size in mobile devices has a similar constraint.

Fresh parameters can effectively infer the DNN task with the satisfied performance. To measure the freshness of the DNN service parameters, we introduce the concept of AoI to quantify the age in the MEC server m:

$$\Delta_{m,j}(t) = t - t_j, \tag{5}$$

where t_j is the timeslot of the latest periodical training of service type j. The same calculation of $\Delta_{n,j}(t)$ is in mobile devices. The updating mechanism at the MEC server m can be defined as $\Delta_{m,j}(t) = \frac{d^{\text{par}}_j}{r_{0,m}(t)}$ if fetching ends at timeslot t; otherwise, $\Delta_{m,j}(t) = \Delta_{m,j}(t-1) + 1$, and at mobile device n: $\Delta_{n,j}(t) = \frac{d^{\text{par}}_j}{r_{0,n}(t)}$ if fetching ends at timeslot t; otherwise, $\Delta_{n,j}(t) = \Delta_{n,j}(t-1) + 1$. Since the DNN is trained periodically in the application server, the DNN training round of type j can be denoted as T^{int}_j. If the AoI of the parameters is less than the training round, the parameters can be regarded as fresh parameters. Let $x^{\text{fre}}_{m,j}(t)$ be the service parameter freshness status, defined as follows: $x^{\text{fre}}_{m,j}(t) = 1$ if $\Delta_{m,j}(t) < T^{\text{int}}_j$; otherwise, $x^{\text{fre}}_{m,j}(t) = 0$, and $x^{\text{fre}}_{n,j}(t) = 1$ if $\Delta_{n,j}(t) < T^{\text{int}}_j$; otherwise, $x^{\text{fre}}_{n,j}(t) = 0$. If $x^{\text{fre}}_{m,j}(t) = 0$

or $x^{\text{fre}}_{n,j}(t) = 0$, the MEC server or the mobile device is required to fetch an up-to-date version of the service parameters from the application server; the fetching times are $T^{\text{par}}_{0,m,j}$ and $T^{\text{par}}_{0,n,j}$, respectively.

2.4. Execution Model

In terms of execution, the inference task is executed under the existence of the corresponding service. If there is no DNN service caching at the MEC server or mobile device, they are required to fetch a DNN service cache and then further carry out the execution. After fetching the service caching, the execution delay of inference task $k_n(t)$ at the MEC server m is calculated as

$$T^{\text{exe}}_{m,k_n(t)} = x^{\text{exe}}_{m,k_n(t)}(t)\frac{c_{k_n(t)}}{f_m(t)}, \tag{6}$$

where $f_m(t)$ is the computation capability of the MEC server m. Likewise, the execution delay at the mobile device n is calculated as

$$T^{\text{exe}}_{n,k_n(t)} = (1 - \sum_{m\in\mathcal{M}} x^{\text{exe}}_{m,k_n(t)}(t))\frac{c_{k_n(t)}}{f_n(t)}, \tag{7}$$

where $f_n(t)$ is the constant computation capability of the mobile device n. Here, the computation capability of a mobile server is less than a MEC server, and $f_m(t)$ has an upper bound: $f_n(t) < f_m(t) \leq f^{\max}$, where $f^{\max}$ is the maximum of the computation capability.

2.5. Energy Model

From the perspective of energy consumption, we focus on the energy of mobile devices since they usually have batteries of limited capacity while the MEC server is connected to the power grid. Hence, the energy consumed for the local execution of the mobile device n can be calculated as

$$E^{\text{exe}}_{n,k_n(t)} = (1 - \sum_{m\in\mathcal{M}} x^{\text{exe}}_{m,k_n(t)}(t))\mu c_{k_n(t)} f_n^2(t), \tag{8}$$

where μ refers to the effective switched capacitance.

In the case of offloading, the energy consumption of the mobile device only includes the uploading energy, calculated as

$$E^{\text{upl}}_{n,m,k_n(t)} = x^{\text{exe}}_{m,k_n(t)}(t)p_{n,m}(t)\frac{d_{k_n(t)}}{r_{n,m}(t)}. \tag{9}$$

Energy consumption is another crucial metric of mobile devices. The cost of the mobile device consists of the time delay and energy consumption with distinct emphasis.

2.6. Cost Model

At timeslot t, the mobile device n with the generated DNN inference task $k_n(t)$ can make an offloading decision to process the task. According to the service caching placement decision and service parameter freshness status, the cost of the mobile device can be divided into the following cases, as seen in Figure 1.

2.6.1. Case 1: Offloading with Fresh Cache

First, in case 1, the mobile device offloads the inference task to the MEC server with caching service libraries and fresh parameters. The combination of the decision and status satisfies: $x_{k_n(t),1}(t) = x^{\text{exe}}_{m,k_n(t)}(t)x^{\text{typ}}_{k_n(t),j}x^{\text{cac}}_{m,j}(t)x^{\text{fre}}_{m,j}(t) = 1$. The total time delay, in this case, can be calculated as $T_{k_n(t),1} = T^{\text{upl}}_{n,m,k_n(t)} + T^{\text{exe}}_{m,k_n(t)}$. In addition, the total energy consumption of the mobile device is represented as $E_{k_n(t),1} = E^{\text{upl}}_{n,m,k_n(t)}$.

2.6.2. Case 2: Offloading with Stale Cache

In Case 2, the mobile device offloads the inference task to the MEC server with caching service libraries and stale parameters. The combination of the decision and status satisfies: $x_{k_n(t),2}(t) = x^{\text{exe}}_{m,k_n(t)}(t)x^{\text{typ}}_{k_n(t),j}x^{\text{cac}}_{m,j}(t)(1 - x^{\text{fre}}_{m,j}(t)) = 1$. The total time delay, in this case, can be calculated as $T_{k_n(t),2} = T^{\text{upl}}_{n,m,k_n(t)} + T^{\text{par}}_{0,m,j} + T^{\text{exe}}_{m,k_n(t)}$. Moreover, the total energy consumption of the mobile device is denoted as $E_{k_n(t),2} = E^{\text{upl}}_{n,m,k_n(t)}$.

2.6.3. Case 3: Offloading without Cache

Then, in case 3, the mobile device offloads the inference task to the MEC server without any DNN service cache. The combination of the decision and status satisfies $x_{k_n(t),3}(t) = x^{\text{exe}}_{m,k_n(t)}(t)x^{\text{typ}}_{k_n(t),j}(1 - x^{\text{cac}}_{m,j}(t)) = 1$. The total time delay, in this case, can be calculated as follows: $T_{k_n(t),3} = T^{\text{upl}}_{n,m,k_n(t)} + T^{\text{lib}}_{0,m,j} + T^{\text{par}}_{0,m,j} + T^{\text{exe}}_{m,k_n(t)}$. Likewise, the total energy consumption of the mobile device is also represented as $E_{k_n(t),3} = E^{\text{upl}}_{n,m,k_n(t)}$.

2.6.4. Case 4: Local Execution with Fresh Cache

For local execution, in case 4, the mobile device locally executes the inference task with caching service libraries and fresh parameters. The combination of the decision and status satisfies $x_{k_n(t),4}(t) = (1 - \sum_{m\in\mathcal{M}} x^{\text{exe}}_{m,k_n(t)}(t))x^{\text{typ}}_{k_n(t),j}x^{\text{cac}}_{n,j}(t)x^{\text{fre}}_{n,j}(t) = 1$. The total time delay, in this case, can be calculated as follows: $T_{k_n(t),4} = T^{\text{exe}}_{n,k_n(t)}$. In addition, the total energy consumption of the mobile device is denoted as $E_{k_n(t),4} = E^{\text{exe}}_{n,k_n(t)}$.

2.6.5. Case 5: Local Execution with Stale Cache

In case 5, the mobile device locally executes the inference task with caching service libraries and stale parameters. The combination of the decision and status satisfies $x_{k_n(t),5}(t) = (1 - \sum_{m\in\mathcal{M}} x^{\text{exe}}_{m,k_n(t)}(t))x^{\text{typ}}_{k_n(t),j}x^{\text{cac}}_{n,j}(t)(1 - x^{\text{fre}}_{n,j}(t)) = 1$. The total time delay, in this case, can be calculated as $T_{k_n(t),5} = T^{\text{par}}_{0,n,j} + T^{\text{exe}}_{n,k_n(t)}$. Then, the total energy consumption of the mobile device is calculated as $E_{k_n(t),5} = E^{\text{exe}}_{n,k_n(t)}$.

2.6.6. Case 6: Local Execution without Cache

Finally, in case 6, the mobile device locally executes the inference task without any DNN service cache. The combination of the decision and status satisfies $x_{k_n(t),6}(t) = (1 - \sum_{m\in\mathcal{M}} x^{\text{exe}}_{m,k_n(t)}(t))x^{\text{typ}}_{k_n(t),j}(1 - x^{\text{cac}}_{n,j}(t)) = 1$. The total time delay, in this case, can be calculated as $T_{k_n(t),6} = T^{\text{lib}}_{0,n,j} + T^{\text{par}}_{0,n,j} + T^{\text{exe}}_{n,k_n(t)}$. Similarly, the total energy consumption of the mobile device is also denoted as $E_{k_n(t),6} = E^{\text{exe}}_{n,k_n(t)}$.

3. Problem Formulation

In the AoI-aware caching-assisted asymmetric offloading scenario, the average cost of the mobile device and the total bandwidth between the application server and MEC servers or mobile devices should be considered due to their crucial effectiveness. On the one hand, minimizing the average cost of the mobile device can ensure that the real-time requirements of the generated inference tasks are met and the battery energy is conserved. As the consumed bandwidth of the application server is limited, while it bears other real-time inference tasks, it is required to minimize the total bandwidth consumption between the application server and MEC servers or mobile devices. Accordingly, the average cost of time completion delay is as follows:

$$T^{\text{ave}} = \frac{1}{|\mathcal{T}||\mathcal{N}|}\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}}\Big(\sum_{m\in\mathcal{M}}\sum_{i=1}^{3} x_{k_n(t),i}(t)T_{k_n(t),i} + \sum_{i=4}^{6} x_{k_n(t),i}(t)T_{k_n(t),i}\Big), \tag{10}$$

where $x_{k_n(t),i}(t)$ is defined as $x_{k_n(t),i}(t) = 1$ if $k_n(t)$ is executed via case i; otherwise, $x_{k_n(t),i}(t) = 0$, and satisfies that each inference task must be executed via one of the cases at one MEC server or local mobile device: $\sum_{m\in\mathcal{M}}\sum_{i=1}^{3} x_{k_n(t),i}(t) + \sum_{i=4}^{6} x_{k_n(t),i}(t) = 1$. Then, the average cost of energy consumption can be denoted as

$$E^{\text{ave}} = \frac{1}{|\mathcal{T}||\mathcal{N}|}\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}}\Big(\sum_{m\in\mathcal{M}}\sum_{i=1}^{3} x_{k_n(t),i}(t)E_{k_n(t),i} + \sum_{i=4}^{6} x_{k_n(t),i}(t)E_{k_n(t),i}\Big), \tag{11}$$

the time average global allocation bandwidth between the application server and MEC servers or mobile devices is denoted as b_0.

Therefore, we formally formulate the original problem to minimize the time average global allocation bandwidth and the average cost of the mobile device consisting of inference task completion delay and energy consumption:

$$\min_{x_{k_n(t),i}(t), f_m(t), p_{n,m}(t)} Z = \xi^{\text{tim}}T^{\text{ave}} + \xi^{\text{ene}}E^{\text{ave}} + \xi^{\text{ban}}b_0 \tag{12}$$

$$s.t. \quad T_{k_n(t),i} \le T^{\max}_{k_n(t)}, \forall n \in \mathcal{N}, t \in \mathcal{T}, i \in \{1, \cdots, 6\}, \tag{13}$$

$$E_{k_n(t),i} \le E^{\max}_{k_n(t)}, \forall n \in \mathcal{N}, t \in \mathcal{T}, i \in \{1, \cdots, 6\}, \tag{14}$$

$$\sum_{m\in\mathcal{M}}\sum_{i=1}^{3} x_{k_n(t),i}(t) + \sum_{i=4}^{6} x_{k_n(t),i}(t) = 1, \tag{15}$$

$$x^{\text{exe}}_{m,k_n(t)}(t) \in \{0,1\}, \forall m \in \mathcal{M}, n \in \mathcal{N}, t \in \mathcal{T}, \tag{16}$$

$$x^{\text{cac}}_{m,j}(t) \in \{0,1\}, \forall m \in \mathcal{M}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{17}$$

$$x^{\text{fre}}_{m,j}(t) \in \{0,1\}, \forall m \in \mathcal{M}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{18}$$

$$\sum_{j\in\mathcal{J}} x^{\text{cac}}_{m,j}(t)(d^{\text{lib}}_j + d^{\text{par}}_j) \le d^{\max}_m, \forall m \in \mathcal{M}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{19}$$

$$\sum_{j\in\mathcal{J}} x^{\text{cac}}_{n,j}(t)(d^{\text{lib}}_j + d^{\text{par}}_j) \le d^{\max}_n, \forall n \in \mathcal{N}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{20}$$

$$f_n(t) < f_m(t) \le f^{\max}, \forall m \in \mathcal{M}, n \in \mathcal{N}, t \in \mathcal{T}, \tag{21}$$

$$p_{n,m}(t) \le p^{\max}, \forall m \in \mathcal{M}, n \in \mathcal{N}, t \in \mathcal{T}, \tag{22}$$

where $x_{k_n(t),i}(t)$, $f_m(t)$, and $p_{n,m}(t)$ are optimization variables. ξ^{ban}, ξ^{tim}, and ξ^{ene} are the given weights of the average global AoI, average time cost, and average energy cost, respectively. (13) and (14) indicate that the inference task completion time delay and consumed energy have upper bounds. According to (15), each inference task has to be executed via (at most) one case at one MEC server or local mobile device. (16)–(18) show that the optimization variables are binary. (19) and (20) constrain the caching capacity limit of heterogeneous services at the MEC server or the mobile device. (21) shows that the computation capability of the MEC server is higher than the mobile device and has a maximum. (22) restricts the upper bound of the uplink transmission power of the mobile device.

$x^{\text{exe}}_{m,k_n(t)}(t)$, $x^{\text{cac}}_{m,j}(t)$, and $x^{\text{fre}}_{m,j}(t)$ are discrete binary integer variables; $p_{n,m}(t)$ and $f_m(t)$ are continuous variables. The objective functions are not linear to the variables, which are coupled mutually. Therefore, problem (12) is an MINLP problem known as NP-hard. It is difficult to solve the problem within the polynomial time. Combining the practical asymmetric environment, it is more challenging to analyze and propose a solution.

3.1. Average Cost Minimization Problem

From the perspective of mobile devices, we first decompose problem (12) into a problem to minimize the average cost of mobile devices:

$$\min_{x_{k_n(t),i}(t), f_m(t), p_{n,m}(t)} \xi^{\text{tim}} T^{\text{ave}} + \xi^{\text{ene}} E^{\text{ave}} \tag{23}$$

$$s.t. \quad \text{(C1)–(C10)},$$

where mobile devices make their decisions based on the weighted sum of time delay and energy consumption.

3.2. Bandwidth Consumption Minimization Problem

From the perspective of the application server, the total bandwidth allocated to the requested MEC server or mobile device is constrained when it transmits the requested service data. We secondly decompose problem (12) into a problem minimizing the consumed bandwidth of the application server:

$$\min \xi^{\text{ban}} b_0 \tag{24}$$

$$s.t. \quad b_0 \leq \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} b_0^{\max}, \tag{25}$$

where $b_0^{\max}$ is the total allocated bandwidth upper bound of the application server at one timeslot.

3.3. Service Fetching Time Minimization Problem

When the application server transmits the service data to the MEC servers or mobile devices, the total transmission time of the responding service data can be minimized based on the total allocated bandwidth. We further formulate problem (26):

$$\min \ T^{\text{fet}}(t) \tag{26}$$

$$s.t. \quad b_{0,m}(t) + \sum_{n \in \mathcal{N}(t)} b_{0,n}(t) \leq b_0(t), \tag{27}$$

$$x_{k_n(t),2}(t) + x_{k_n(t),3}(t) + x_{k_n(t),5}(t) + x_{k_n(t),6}(t) = 1, \tag{28}$$

$$x_{k_n(t),i}(t) \in \{0,1\}, i = \{2,3,5,6\}, \tag{29}$$

where $T^{\text{fet}}(t)$ is the total transmission time of the responding service data. (27) indicates that the total allocated bandwidth has an upper bound, (28) and (29) limit the combination decisions.

Here, we clarify the connections among these three subproblems and how they can work together to reach the optimal solution for problem (12). Problem (12) jointly minimizes the cost of mobile devices and the global allocation bandwidth. Firstly, problem (23) minimizes the mobile device cost, including the time delay and energy consumption. Secondly, problem (24) minimizes the time average allocation bandwidth from a global perspective. Thirdly, problem (26) further minimizes the responding service transmission time after making the offloading decision based on the solution of the problem (23).

4. Solution

In this section, we propose three modules to, respectively, solve the subproblems in the last section. In particular, the LMKO module can minimize the average cost of mobile devices. To minimize the consumed bandwidth of the application server, we devise the LLUC module. Moreover, the KCDF module minimizes the total transmission time of the responding service data.

4.1. Method of Lagrange Multipliers with the KKT Condition-Based Offloading Module (LMKO)

To minimize the average cost of mobile devices, we transform problem (23) into a problem of tractable form and further leverage convex optimization to solve it. According to constraint (13), the time delay of each case cannot exceed the inference task completion tolerance deadline. Hence, we set the time delay of the case with most of the procedures to the maximum tolerance time to reduce the number of optimization variables: $T_{k_n(t),3} = T^{\max}_{k_n(t)}$. For succinct expression, we define notations A and B as

$$A = T^{\max}_{k_n(t)} - T^{\text{lib}}_{0,m,j} - T^{\text{par}}_{0,m,j}, \tag{30}$$

$$B = \frac{\sigma^2 + I_{n,m}(t)}{h_{n,m}(t)}. \tag{31}$$

Then, $f_m(t)$ and $p_{n,m}(t)$ can be transform into the function values of $g_1(T^{\text{exe}}_{m,k_n(t)}(t))$ and $g_2(T^{\text{exe}}_{m,k_n(t)}(t))$, respectively:

$$f_m(t) = g_1(T^{\text{exe}}_{m,k_n(t)}(t)) = \frac{c_{k_n(t)}}{T^{\text{exe}}_{m,k_n(t)}(t)}, \tag{32}$$

$$p_{n,m}(t) = g_2(T^{\text{exe}}_{m,k_n(t)}(t)) = (2^{\frac{d_{k_n(t)}}{b_{n,m}(t)(A - T^{\text{exe}}_{m,k_n(t)})}} - 1)B. \tag{33}$$

Moreover, the cost of the objective function in problem (23) can be calculated as

$$\begin{aligned} Z_1 \quad &= \xi^{\text{tim}} \tfrac{1}{|\mathcal{T}||\mathcal{N}|} \sum_{t\in\mathcal{T}} \sum_{n\in\mathcal{N}} \Big(\sum_{m\in\mathcal{M}} (x_{k_n(t),1}(t)A \\ &+ x_{k_n(t),2}(t)(T^{\max}_{k_n(t)} - T^{\text{lib}}_{0,m,j}) \\ &+ x_{k_n(t),3}(t)T^{\max}_{k_n(t)}) + \sum_{i=4}^{6} x_{k_n(t),i}(t)T_{k_n(t),i}) \\ &+ \xi^{\text{ene}} \tfrac{1}{|\mathcal{T}||\mathcal{N}|} \sum_{t\in\mathcal{T}} \sum_{n\in\mathcal{N}} \Big(\sum_{m\in\mathcal{M}} \sum_{i=1}^{3} x_{k_n(t),i}(t) \\ &\times g_2(T^{\text{exe}}_{m,k_n(t)}(t))(A - T^{\text{exe}}_{m,k_n(t)}(t)) \\ &+ \sum_{i=4}^{6} x_{k_n(t),i}(t)E_{k_n(t),i}). \end{aligned} \tag{34}$$

Therefore, problem (23) can be further transformed into:

$$\min_{T^{\text{exe}}_{m,k_n(t)}(t), x_{k_n(t),i}(t)} Z_1 \tag{35}$$

$$s.t. \quad g_1(T^{\text{exe}}_{m,k_n(t)}(t)) \le f^{\max}, \tag{36}$$

$$g_2(T^{\text{exe}}_{m,k_n(t)}(t)) \le p^{\max}, \tag{37}$$

$$x_{k_n(t),i}(t) \in [0,1], \tag{38}$$

where $T^{\text{exe}}_{m,k_n(t)}(t)$ and $x_{k_n(t),i}(t)$ are optimization variables. Constraint (36) reflects the computation capability limit and $T^{\text{exe}}_{m,k_n(t)}(t)$. (37) constrains the relationship between the maximum power and $T^{\text{exe}}_{m,k_n(t)}(t)$. (38) indicates that the decision combination is relaxed to be continuous.

Subsequently, we leverage the method using Lagrange multipliers with KKT conditions [25] to solve problem (35). Before this, we prove that the problem (35) is convex.

Now, we define another function of $T^{\text{exe}}_{m,k_n(t)}(t)$ as follows:

$$g_3(T^{\text{exe}}_{m,k_n(t)}(t)) = g_2(T^{\text{exe}}_{m,k_n(t)}(t))(A - T^{\text{exe}}_{m,k_n(t)}(t)), \tag{39}$$

and take the second partial derivative of $g_3(T^{\text{exe}}_{m,k_n(t)}(t))$ with respect to $T^{\text{exe}}_{m,k_n(t)}(t)$:

$$\frac{\partial^2 g_3}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} = \frac{\ln 2^2 {d_{k_n(t)}}^2 B 2^{\frac{d_{k_n(t)}}{b_{n,m}(t)(A - T^{\text{exe}}_{m,k_n(t)}(t))}}}{b_{n,m}(t)^2 (A - T^{\text{exe}}_{m,k_n(t)}(t))^3}. \tag{40}$$

All of the terms in (40) are positive, $\frac{\partial^2 g_3}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} > 0$, and $g_3(T^{\text{exe}}_{m,k_n(t)}(t))$ is a convex function. Similarly, $g_3(x_{k_n(t),i}(t) T^{\text{exe}}_{m,k_n(t)}(t))$ is a convex function. Furthermore, we define a perspective function of $g_3(x_{k_n(t),i}(t) T^{\text{exe}}_{m,k_n(t)}(t))$ as

$$\begin{aligned} & g_4(x_{k_n(t),i}(t) T^{\text{exe}}_{m,k_n(t)}(t), x_{k_n(t),i}(t)) \\ = \; & x_{k_n(t),i}(t) g_3(\tfrac{x_{k_n(t),i}(t) T^{\text{exe}}_{m,k_n(t)}(t)}{x_{k_n(t),i}(t)}) \\ = \; & x_{k_n(t),i}(t) g_2(T^{\text{exe}}_{m,k_n(t)}(t))(A - T^{\text{exe}}_{m,k_n(t)}(t)). \end{aligned} \tag{41}$$

According to (41), $g_4(x_{k_n(t),i}(t) T^{\text{exe}}_{m,k_n(t)}(t), x_{k_n(t),i}(t))$ is convex so that the objective function of the problem (35) is a convex function. Then, the second partial derivatives of $g_1(T^{\text{exe}}_{m,k_n(t)}(t))$ and $g_2(T^{\text{exe}}_{m,k_n(t)}(t))$ with respect to $T^{\text{exe}}_{m,k_n(t)}(t)$ are, respectively, calculated as (42) and (43):

$$\frac{\partial^2 g_1}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} = \frac{2 C_{k_n(t)}}{{T^{\text{exe}}_{m,k_n(t)}(t)}^3}, \tag{42}$$

$$\begin{aligned} \frac{\partial^2 g_2}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} = \; & \frac{\ln 2 d_{k_n(t)} B 2^{\frac{d_{k_n(t)}}{b_{n,m}(t)(A - T^{\text{exe}}_{m,k_n(t)}(t))}}}{b_{n,m}(t)(A - T^{\text{exe}}_{m,k_n(t)}(t))^3} \\ & \times (\frac{\ln 2 d_{k_n(t)}}{b_{n,m}(t)(A - T^{\text{exe}}_{m,k_n(t)}(t))} + 2). \end{aligned} \tag{43}$$

All terms in (42) and (43) are positive, $\frac{\partial^2 g_1}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} > 0$, and $\frac{\partial^2 g_2}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} > 0$. Thus, constraint (36) and (37) are convex with $T^{\text{exe}}_{m,k_n(t)}(t)$. The feasible region of the Problem (35) is a convex set. We can derive that the problem (35) is convex. In addition, if $p^{\max}$ and $f^{\max}$ are high enough, we can find a feasible solution to making all of the constraints slack, hence satisfying the Slater condition. A convex problem that satisfies the Slater condition is sufficient for the problem and its dual problem to be strong. In other words, they have zero dual gap and their optimal solutions are equal.

Next, we define the Lagrangian relaxation function of the problem (35) as

$$\begin{aligned} & L(T^{\text{exe}}_{m,k_n(t)}(t), x_{k_n(t),i}(t), \lambda_{m,k_n(t),1}, \lambda_{m,k_n(t),2}) \\ = & C + \sum_{t \in \mathcal{T}} \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} (\lambda_{m,k_n(t),1}(g_1(T^{\text{exe}}_{m,k_n(t)}(t)) - f^{\max}) \\ & + \lambda_{m,k_n(t),2}(g_2(T^{\text{exe}}_{m,k_n(t)}(t)) - p^{\max})), \end{aligned} \tag{44}$$

where $\lambda_{m,k_n(t),1}$ and $\lambda_{m,k_n(t),2}$ are the Lagrangian multipliers. The Lagrangian relaxation function relaxes the constraints of the Problem (35). Here, we formally transform Problem (35) into its dual problem:

$$\max_{\lambda_{m,k_n(t),1}, \lambda_{m,k_n(t),2}} \min_{T^{\text{exe}}_{m,k_n(t)}(t), x_{k_n(t),i}(t)} L \tag{45}$$

$$s.t. \quad \lambda_{m,k_n(t),1} \geq 0, \; \lambda_{m,k_n(t),2} \geq 0, \tag{46}$$

where we first fix $\lambda_{m,k_n(t),1}$, $\lambda_{m,k_n(t),2}$ and minimize L to obtain the infimum, then fix $T^{\text{exe}}_{m,k_n(t)}(t)$, $x_{k_n(t),i}(t)$ and maximize the infimum. (46) indicates that the Lagrangian multipliers are positive.

We further detail the KKT condition of the problem (45):

$$\begin{aligned}\frac{\partial L}{\partial T^{\text{exe}}_{m,k_n(t)}} = & \frac{\zeta^{\text{ene}}}{|\mathcal{T}||\mathcal{N}|}\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}} x_{k_n(t),i}(t)\frac{\partial g_3}{\partial T^{\text{exe}}_{m,k_n(t)}} \\ & + \sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\lambda_{m,k_n(t),1}\frac{\partial g_1}{\partial T^{\text{exe}}_{m,k_n(t)}} \\ & +\lambda_{m,k_n(t),2}\frac{\partial g_2}{\partial T^{\text{exe}}_{m,k_n(t)}}) = 0,\end{aligned} \tag{47}$$

$$\lambda_{m,k_n(t),1}(g_1(T^{\text{exe}}_{m,k_n(t)}(t)) - f^{\max}) = 0, \tag{48}$$

$$\lambda_{m,k_n(t),2}(g_2(T^{\text{exe}}_{m,k_n(t)}(t)) - p^{\max}) = 0, \tag{49}$$

$$g_1(T^{\text{exe}}_{m,k_n(t)}(t)) \le f^{\max}, \tag{50}$$

$$g_2(T^{\text{exe}}_{m,k_n(t)}(t)) \le p^{\max}, \tag{51}$$

$$\lambda_{m,k_n(t),1} \ge 0,\ \lambda_{m,k_n(t),2} \ge 0, \tag{52}$$

where

$$\frac{\partial g_3}{\partial T^{\text{exe}}_{m,k_n(t)}} = B\left(\frac{\ln 2 d_{k_n(t)} 2^{\frac{d_{k_n(t)}}{b_{n,m}(t)(A-T^{\text{exe}}_{m,k_n(t)})}}}{b_{n,m}(t)(A-T^{\text{exe}}_{m,k_n(t)})} - 2^{\frac{d_{k_n(t)}}{b_{n,m}(t)(A-T^{\text{exe}}_{m,k_n(t)})}} + 1\right), \tag{53}$$

$$\frac{\partial g_1}{\partial T^{\text{exe}}_{m,k_n(t)}} = -\frac{c_{k_n(t)}}{{T^{\text{exe}}_{m,k_n(t)}}^2}, \tag{54}$$

and

$$\frac{\partial g_2}{\partial T^{\text{exe}}_{m,k_n(t)}} = \frac{B \ln 2 d_{k_n(t)} 2^{\frac{d_{k_n(t)}}{b_{n,m}(t)(A-T^{\text{exe}}_{m,k_n(t)})}}}{b_{n,m}(t)(A-T^{\text{exe}}_{m,k_n(t)})^2}. \tag{55}$$

In KKT conditions, (47) is the dual stationarity condition. (48) and (49) are the complementary slackness conditions. (50) and (51) are the primal feasibility conditions while (52) is the dual feasibility condition.

Since (47) is a transcendental equation, we derive the solution by the Newton iteration method:

$${T^{\text{exe}}_{m,k_n(t)}}' = T^{\text{exe}}_{m,k_n(t)} - \frac{\frac{\partial L}{\partial T^{\text{exe}}_{m,k_n(t)}}}{\frac{\partial^2 L}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2}}, \tag{56}$$

where

$$\begin{aligned}\frac{\partial^2 L}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} = & \frac{\zeta^{\text{ene}}}{|\mathcal{T}||\mathcal{N}|}\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}} x_{k_n(t),i}(t)\frac{\partial^2 g_3}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} \\ & + \sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\lambda_{m,k_n(t),1}\frac{\partial^2 g_1}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2} + \lambda_{m,k_n(t),2}\frac{\partial^2 g_2}{\partial {T^{\text{exe}}_{m,k_n(t)}}^2}) = 0.\end{aligned} \tag{57}$$

After iterations of the Newton method, we obtain the optimal solution ${T^{\text{exe}}_{m,k_n(t)}}^*$. Then, the optimal resource allocation $f_m(t)^*$ and $p_{n,m}(t)^*$ can be calculated according to (32) and (33), respectively.

Moreover, we leverage the subgradient method to update the Lagrangian multipliers:

$$\begin{aligned}{\lambda_{m,k_n(t),1}}' = & \max\{\lambda_{m,k_n(t),1} \\ & +\alpha_{m,k_n(t),1}(g_1(T^{\text{exe}}_{m,k_n(t)}(t)) - f^{\max}), 0\},\end{aligned} \tag{58}$$

$$\begin{aligned}{\lambda_{m,k_n(t),2}}' = & \max\{\lambda_{m,k_n(t),2} \\ & +\alpha_{m,k_n(t),2}(g_2(T^{\text{exe}}_{m,k_n(t)}(t)) - p^{\max}), 0\},\end{aligned} \tag{59}$$

where $\alpha_{m,k_n(t),1}$ and $\alpha_{m,k_n(t),2}$ are the diminishing step size, respectively.

Since problem (45) is convex with respect to the optimization variable, the update iteration can converge to the optimal solution, satisfying the following conditions: $\sum_{\tau^{\text{sub}}=1}^{\infty} \alpha_{m,k_n(t),1}(\tau^{\text{sub}}) = \infty$, $\sum_{\tau^{\text{sub}}=1}^{\infty} \alpha_{m,k_n(t),2}(\tau^{\text{sub}}) = \infty$, $\sum_{\tau^{\text{sub}}=1}^{\infty} \alpha_{m,k_n(t),1}(\tau^{\text{sub}})^2 < \infty$, and $\sum_{\tau^{\text{sub}}=1}^{\infty} \alpha_{m,k_n(t),2}(\tau^{\text{sub}})^2 < \infty$, where τ^{sub} is the iteration index. There is proof in [26].

Based on the given service caching placement decision and service parameter freshness status, we can calculate the cost of local execution as follows:

$$C^{\text{loc}}_{k_n(t)} = x_{k_n(t),i}(t)(\xi^{\text{tim}} T^*_{k_n(t)} + \xi^{\text{ene}} E^*_{k_n(t)}), i \in \{1,2,3\}, \tag{60}$$

the offloading cost of the MEC server m is calculated as

$$C^{\text{off}}_{m,k_n(t)} = x_{k_n(t),i}(t)(\xi^{\text{tim}} T^*_{k_n(t)} + \xi^{\text{ene}} E^*_{k_n(t)}), i \in \{4,5,6\}, \tag{61}$$

and the minimum offloading cost among all the MEC servers is calculated as

$$C^{\text{off}}_{m^*,k_n(t)} = \min_{m \in \mathcal{M}} C^{\text{off}}_{m,k_n(t)} \tag{62}$$

where $T^*_{k_n(t)}$ and $E^*_{k_n(t)}$ are calculated according to $f_m(t)^*$ and $p_{n,m}(t)^*$. The offloading decision can be further derived. If $C^{\text{loc}}_{k_n(t)} < C^{\text{off}}_{m^*,k_n(t)}$, $x^{\text{exe}}_{m^*,k_n(t)}(t) = 1$, and $x^{\text{exe}}_{m,k_n(t)}(t) = 0$, $\forall m \in \mathcal{M} \backslash m^*$, the inference task is offloaded to the MEC server m^*; otherwise, $x^{\text{exe}}_{m,k_n(t)}(t) = 0$, $\forall m \in \mathcal{M}$, and the inference task is executed locally.

The pseudo-code of the LMKO is shown in Algorithm 1. The complexity of LMKO is $O(|\mathcal{M}|(\tau^{\text{new}} + \tau^{\text{sub}})) + |\mathcal{M}|)$, where τ^{new} and τ^{sub} are the iteration numbers of the Newton method and the subgradient method, respectively.

4.2. Lyapunov Optimization-Based Learning and Update Control Module (LLUC)

Since the application server also bears other applications, its bandwidth resources are limited and need to be economized. In this subsection, we minimize the bandwidth consumption of the application server from the perspective of a global view while minimizing the service fetching time to accelerate the inference task processing.

The inference tasks are generated randomly, and the execution request in the offloading style or local style is a random event for the MEC server and mobile device. If there is no caching service or fresh service parameters, they call for the application server to fetch the service. Therefore, the fetching request is also random in terms of the application server, which has no a priori distribution. We regard the total requested service data size waiting for transmission as a queue, and leverage the Lyapunov optimization to solve the problem of stabilizing a randomly arriving queue system.

The application server transmits the requested service data as soon as possible to decrease the fetching time. At timeslot t, the total requested service data size can be defined as the enqueued rate:

$$\begin{aligned} d^{\text{enq}}(t) = \quad & \sum_{n \in \mathcal{N}} \Big(\sum_{m \in \mathcal{M}} (x_{k_n(t),2}(t) d^{\text{par}}_j \\ & + x_{k_n(t),3}(t)(d^{\text{lib}}_j + d^{\text{par}}_j)) + x_{k_n(t),5}(t) d^{\text{par}}_j + x_{k_n(t),6}(t)(d^{\text{lib}}_j + d^{\text{par}}_j)). \end{aligned} \tag{63}$$

Moreover, let d^{deq} be the dequeued rate, which is the total size of the service transmitted from the application server to the requesting MEC servers or mobile devices. Furthermore, the backlog of the queue can be defined as $Q(t+1) = \max\{Q(t) + d^{\text{enq}}(t) - d^{\text{deq}}(t), 0\}$, where the enqueued rate and dequeued rate can affect the queue backlog of the next timeslot. Then, we define the quadratic Lyapunov function as $Y(t) = \frac{Q(t)^2}{2}$, and the Lyapunov drift can be denoted as $\Delta Y(t) = Y(t+1) - Y(t)$. In addition, we define the penalty function of the Lyapunov optimization, which equals the total allocated bandwidth consumption for transmitting the requested service data at timeslot t: $b_0(t) = \beta d^{\text{deq}}(t)$, where β is the simplified transformation coefficient.

We formally transform problem (24) into the Lyapunov optimization problem:

$$\min_{d^{\text{deq}}(t)} Z_2 = \Delta Y(t) + V(t) b_0(t) \tag{64}$$

$$s.t. \quad \lim_{t\to\infty} \frac{Q(t)}{t} = 0, \tag{65}$$

$$b_0(t) \le b_0^{\max}, \tag{66}$$

where $V(t)$ is the adaptive weight of the penalty, and (65) is the stable condition of the queue system, and $b_0^{\max}$ in (66) is the maximum of the total available bandwidth between the application server and all requesting MEC servers or mobile devices.

Algorithm 1 Method of Lagrange multipliers with the KKT condition-based offloading module (LMKO).

Require: time cost $T_{k_n(t),i}$, energy cost $E_{k_n(t),i}$, time weight ξ^{tim}, delay weight ξ^{ene}, maximum tolerance time $T_{k_n(t)}^{\max}$, maximum computation capability $f^{\max}$, maximum power $p^{\max}$, maximum Newton iteration τ^{new}, subgradient threshold ϵ^{sub}

Ensure: offloading decision $x_{m,k_n(t)}^{\text{exe}}(t)$

for $t = 1$ to $|\mathcal{T}|$ **do**
  for $t = 1$ to $|\mathcal{N}|$ **do**
    for $t = 1$ to $|\mathcal{M}|$ **do**
      for $t = 1$ to τ^{new} **do**
        Calculate ${T_{m,k_n(t)}^{\text{exe}}}'$ based on the Newton iteration method according to (56).
      end for
      Obtain ${T_{m,k_n(t)}^{\text{exe}}}^{*}$.
      repeat
        Update ${\lambda_{m,k_n(t),1}}'$ and ${\lambda_{m,k_n(t),2}}'$ based on the subgradient method according to (58) and (59), respectively.
      until ${\lambda_{m,k_n(t),1}}' - \lambda_{m,k_n(t),1} \le \epsilon^{\text{sub}}$ and ${\lambda_{m,k_n(t),2}}' - \lambda_{m,k_n(t),2} \le \epsilon^{\text{sub}}$
      Calculate ${f_m(t)}^{*}$, ${p_{n,m}(t)}^{*}$, and $C_{m,k_n(t)}^{\text{off}}$ according to (32), (33), and (61), respectively.
    end for
    Calculate $C_{k_n(t)}^{\text{loc}}$ and $C_{m^*,k_n(t)}^{\text{off}}$ according to (60) and (62), respectively.
    if $C_{m^*,k_n(t)}^{\text{off}} > C_{k_n(t)}^{\text{loc}}$ **then**
      $x_{m^*,k_n(t)}^{\text{exe}}(t) = 1$.
      $x_{m,k_n(t)}^{\text{exe}}(t) = 0,\ \forall m \in \mathcal{M} \backslash m^*$.
    else
      $x_{m,k_n(t)}^{\text{exe}}(t) = 0,\ \forall m \in \mathcal{M}$.
    end if
  end for
end for
return $x_{m,k_n(t)}^{\text{exe}}(t)$

Theorem 1 ([27]). *Assuming there are constants $D \ge 0$, $\epsilon^{que} > 0$, $V^{max} \ge 0$, $b_0^{max} > 0$, such that for all t and all possible variables $Q(t)$, the Lyapunov drift-plus-penalty condition holds that:*

$$\mathbb{E}[\Delta Y(t) + V(t) b_0(t) | Q(t)] \le D - \epsilon^{que} Q(t) + V^{max} b_0^{max}, \tag{67}$$

where $\mathbb{E}[\frac{1}{2}(d^{enq}(t) - d^{deq}(t))^2 | Q(t)] \le D$ indicates that the difference between the enqueued rate and the dequeued rate has an upper bound, $\mathbb{E}[d^{enq}(t) - d^{deq}(t) | Q(t)] \le -\epsilon^{que}$ indicates that the queue is controlled, V^{max} is the maximum of $V(t)$ over time, and b_0^{max} is the maximum of $b_0(t)$

mentioned above. For all $t > 0$, the time average queue backlog and the time average bandwidth satisfy the following:

$$\frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\mathbb{E}[Q(t)] \leq \frac{D + V^{max}(b_0^{max} - b_0^{min})}{\epsilon^{que}} + \frac{\mathbb{E}[L(1)]}{|\mathcal{T}|\epsilon^{que}}, \tag{68}$$

$$\frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\mathbb{E}[b_0(t)] \leq \frac{D}{V^{max}} + b_0^{max} + \frac{\mathbb{E}[L(1)]}{|\mathcal{T}|V^{max}}, \tag{69}$$

where b_0^{min} is the minimum of $b_0(t)$.

Theorem 1 explains that when the Lyapunov drift-plus-penalty condition is met, the average queue backlog is at most $O(V^{\max})$ complexity, and the average bandwidth is at most $O(\frac{1}{V^{\max}})$ above the maximum bandwidth. Hence, we find that there is a trade-off between the queue backlog and the bandwidth penalty, which is tuned by $V(t)$.

In addition, since $\Delta Y(t) + V(t)b_0(t) \leq \frac{(d^{\text{enq}}(t) - d^{\text{deq}}(t))^2}{2} + Q(t)(d^{\text{enq}}(t) - d^{\text{deq}}(t)) + V(t)\beta d^{\text{deq}}(t)$, we can derive the optimal controlled dequeued rate by taking the derivative with respect to $d^{\text{deq}}(t)$ and further setting it to 0:

$$d^{\text{deq},*}(t) = Q(t) - V(t)\beta + d^{\text{enq}}(t). \tag{70}$$

To improve the Lyapunov optimization, we first design an adaptive learning penalty weight method to adaptively adjust to $V(t)$:

$$V(t) = \frac{V(1)}{e^{\zeta\phi(t)}}, \tag{71}$$

where ζ is the learning rate of penalty weight, $\phi(t) = \frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}(t)}\mathbb{1}\{T_{k_n(t)} > T_{k_n(t)}^{\max}\}$ represents the ratio of the missing tolerance time inference task number, and $\mathbb{1}$ is an indicator function. The emphasis on the bandwidth penalty is lowered as the ratio of the overtime inference tasks increases. When the ratio is alleviated, the weight of the bandwidth is set to be higher.

Secondly, since the transmission data sizes among all the requested MEC servers or mobile devices are distinct, e.g., some request the service libraries and parameters while others only request the parameters, the application server can preferentially respond to request only to the service parameters to decrease the consumption of the bandwidth when the weight of the bandwidth penalty is high. Therefore, we devise a dequeued rate update mechanism. When $V(t) > \epsilon^{\text{thr}}$ where ϵ^{thr} is a given threshold of the penalty weight, the enqueued rate can be updated to:

$$d^{\text{deq},\prime}(t) = \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{J}}(x_{k_n(t),2}(t)d_j^{\text{par}} + x_{k_n(t),5}(t)d_j^{\text{par}}), \tag{72}$$

$$\begin{aligned} d^{\text{deq},\prime}(t+1) = {} & d^{\text{deq},*}(t+1) + \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{J}}(x_{k_n(t),3}(t) \\ & \times(d_j^{\text{lib}} + d_j^{\text{par}}) + x_{k_n(t),6}(t)(d_j^{\text{lib}} + d_j^{\text{par}})), \end{aligned} \tag{73}$$

where the request transmission data sizes of cases 3 and 6 are assigned to timeslot $t+1$ to alleviate the bandwidth penalty at the current timeslot t.

Thirdly, from the perspective of the service parameters with few timeslots until the next training, if the application server directly send the part of data, it can be requested again soon due to its stale service parameters. We further propose a freshness-aware transmitting method to reduce the service-requested frequency; the service parameters that will be trained soon are arranged to be transmitted at the end of their training. If

the time condition satisfies $((t-t_j)\mathrm{mod}T_j^{\mathrm{int}}) > \eta T_j^{\mathrm{int}}$, where η is the given proportion of the training round and mod is an operator of taking the remainder, the dequeued rate is arranged as follows:

$$\begin{aligned} d^{\mathrm{deq},\prime}(t) = \; & d^{\mathrm{deq},*}(t) - \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\sum_{\bar{j}\in\mathcal{J}\setminus j}(x_{k_n(t),2}(t)d_{\bar{j}}^{\mathrm{par}} \\ & + x_{k_n(t),3}(t)(d_{\bar{j}}^{\mathrm{lib}} + d_{\bar{j}}^{\mathrm{par}}) + x_{k_n(t),5}(t)d_{\bar{j}}^{\mathrm{par}} \\ & + x_{k_n(t),6}(t)(d_{\bar{j}}^{\mathrm{lib}} + d_{\bar{j}}^{\mathrm{par}})), \end{aligned} \tag{74}$$

$$\begin{aligned} & d^{\mathrm{deq},\prime}(t + T_j^{\mathrm{int}} - ((t-t_j)\mathrm{mod}T_j^{\mathrm{int}})) \\ = \; & d^{\mathrm{deq},*}(t + T_j^{\mathrm{int}} - ((t-t_j)\mathrm{mod}T_j^{\mathrm{int}})) \\ + \; & \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{J}}(x_{k_n(t),2}(t)d_j^{\mathrm{par}} \\ + \; & x_{k_n(t),3}(t)(d_j^{\mathrm{lib}} + d_j^{\mathrm{par}}) + x_{k_n(t),5}(t)d_j^{\mathrm{par}} \\ + \; & x_{k_n(t),6}(t)(d_j^{\mathrm{lib}} + d_j^{\mathrm{par}})), \end{aligned} \tag{75}$$

where the service ready to be trained is arranged to be transmitted from timeslot t to timeslot $t + T_j^{\mathrm{int}} - ((t-t_j)\mathrm{mod}T_j^{\mathrm{int}})$.

The pseudo-code of LLUC is shown in Algorithm 2. The complexity of the adaptive learning penalty weight method, dequeued rate update mechanism, and freshness-aware transmitting method are $O(|\mathcal{N}|)$, $O(|\mathcal{M}||\mathcal{N}||\mathcal{J}|)$, and $O(|\mathcal{M}||\mathcal{N}||\mathcal{J}|)$, respectively.

Algorithm 2 Lyapunov optimization-based learning and update control module (LLUC).

Require: enqueued rate $d^{\mathrm{enq}}(t)$, initial queue backlog $Q(1)$, transformation coefficient β, initial bandwidth penalty weight $V(1)$, learning rate of penalty weight ζ, given threshold of penalty weight ϵ^{thr}, given training round proportion η
Ensure: dequeued rate $d^{\mathrm{deq},\prime}(t)$

```
for t = 1 to T do
    Update V(t) based on the adaptive learning penalty weight method according to (71).
    Calculate d^{deq,*}(t) based on the Lyapunov optimization according to (70).
    if V(t) > ε^{thr} then
        Update d^{deq,'}(t) and d^{deq,'}(t+1) based on the dequeued rate update mechanism
        according to (72) and (73), respectively.
    end if
    if ((t − t_j) mod T_j^{int}) > η T_j^{int} then
        Update d^{deq,'}(t) and d^{deq,'}(t + T_j^{int} − ((t − t_j) mod T_j^{int})) based on the freshness-
        aware transmitting method according to (74) and (75), respectively.
    end if
end for
return d^{deq,'}(t).
```

4.3. KM Algorithm-Based Channel Division Fetching Module (KCDF)

Since the application server transmits with the MEC servers or mobile devices under the cellular network, the cellular channel matching is crucial to reduce the total transmission time of requested service data $d^{\mathrm{deq},\prime}(t)$.

The total transmission time of the dequeued requested service data can be denoted as

$$\begin{aligned} T^{\mathrm{fet}}(t) = \; & \sum_{n\in\mathcal{N}(t)}(\sum_{m\in\mathcal{M}(t)}(x_{k_n(t),2}(t)T_{0,m,j}^{\mathrm{par}} \\ & + x_{k_n(t),3}(t)(T_{0,m,j}^{\mathrm{lib}} + T_{0,m,j}^{\mathrm{par}})) + x_{k_n(t),5}(t)T_{0,n,j}^{\mathrm{par}} + x_{k_n(t),6}(t)(T_{0,n,j}^{\mathrm{lib}} + T_{0,n,j}^{\mathrm{par}})), \end{aligned} \tag{76}$$

where $\mathcal{M}(t)$ and $\mathcal{N}(t)$ are the responding sets of MEC servers and mobile servers based on the dequeued service data, respectively.

First, we divide the total allocated bandwidth into two parts, one is allocated for transmitting cases 2 and 5, and another is allocated for cases 3 and 6 with more transmitted data. The allocated bandwidth divided method is designed as

$$\begin{aligned} b_0^{\text{par}}(t) = \quad & b_0(t) \sum_{n\in\mathcal{N}(t)} \Big(\sum_{m\in\mathcal{M}(t)} x_{k_n(t),2}(t) d_j^{\text{par}} + x_{k_n(t),5}(t) d_j^{\text{par}} \Big) \\ & / \sum_{n\in\mathcal{N}(t)} \Big(\sum_{m\in\mathcal{M}(t)} (x_{k_n(t),2}(t) d_j^{\text{par}} + x_{k_n(t),3}(t)(d_j^{\text{lib}} + d_j^{\text{par}})) \\ & + x_{k_n(t),5}(t) d_j^{\text{par}} + x_{k_n(t),6}(t)(d_j^{\text{lib}} + d_j^{\text{par}})\Big), \end{aligned} \tag{77}$$

and

$$b_0^{\text{lib,par}}(t) = b_0(t) - b_0^{\text{par}}(t), \tag{78}$$

where $b_0^{\text{par}}(t)$ and $b_0^{\text{lib,par}}(t)$ are the total allocated bandwidths of cases 2 and 5 and cases 3 and 6, based on their total transmitted data sizes, respectively.

Take cases 2 and 5 as an example, we defined the response set as $\mathcal{S} = \mathcal{M}^{\text{par}}(t) \cup \mathcal{N}^{\text{par}}(t)$, where $\mathcal{S}$ is indexed by s and has cardinal number $|\mathcal{S}|$, $\mathcal{M}^{\text{par}}(t)$ and $\mathcal{N}^{\text{par}}(t)$ are the responding sets with cases 2 and 5 of the MEC servers and mobile devices, respectively. Let $\mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$ be the set of $|\mathcal{A}|$ cellular channels indexed by a. The matching decision of s and a can be defined as $x_{s,a}^{\text{mat}}(t) = 1$ if a is allocated to s; otherwise, $x_{s,a}^{\text{mat}}(t) = 0$, and it is constrained by: $\sum_{s\in\mathcal{S}} x_{s,a}^{\text{mat}}(t) = 1, \forall a \in \mathcal{A}$, $\sum_{a\in\mathcal{A}} x_{s,a}^{\text{mat}}(t) = 1, \forall s \in \mathcal{S}$, where each cellular channel is allocated for, at most, one MEC server or mobile device, and each MEC server or mobile device is assigned, at most, one cellular channel. Thus, the transmission latency of the service parameter from the application server to the MEC server or mobile device s over cellular channel a is:

$$T_{s,a}^{\text{par}} = \frac{d_{j(s)}^{\text{par}}}{\frac{b_0^{\text{par}}(t)}{|\mathcal{S}|} \log(1 + \frac{p_{s,a}(t) h_{s,a}(t)}{\sigma^2 + I_{s,a}(t)})}, \tag{79}$$

where $j(s)$ is the service type transmitted for s, $p_{s,a}(t)$, $h_{s,a}(t)$, and $I_{s,a}(t)$ are the transmission power, channel gain, and co-channel interference under channel a to s, respectively.

Here, we formally formulate the problem to minimize the total transmission time of the service parameters:

$$\min_{x_{s,a}^{\text{mat}}(t)} Z_3 = \frac{1}{|\mathcal{S}|} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} x_{s,a}^{\text{mat}}(t) T_{s,a}^{\text{par}} \tag{80}$$

$$s.t. \quad \sum_{s\in\mathcal{S}} x_{s,a}^{\text{mat}}(t) = 1, \forall a \in \mathcal{A}, \tag{81}$$

$$\sum_{a\in\mathcal{A}} x_{s,a}^{\text{mat}}(t) = 1, \forall s \in \mathcal{S}, \tag{82}$$

$$x_{s,a}^{\text{mat}}(t) \in \{0, 1\}, \forall s \in \mathcal{S}, a \in \mathcal{A}, \tag{83}$$

where $x_{s,a}^{\text{mat}}(t)$ is the optimization variable. (81)–(83) are the constraints of the matching decision.

We leverage the KM algorithm [28] to solve the problem. The complete weighted link bipartite graph is defined as $\mathbb{G} = (\mathcal{S}, \mathcal{A}, <\mathcal{S}, \mathcal{A}>)$, where $\mathcal{S}$ and $\mathcal{A}$ are vertex sets of two sides, $<\mathcal{S}, \mathcal{A}>$ is the link set, and the weight of its element is derived from our devised link-initialized method:

$$w_{s,a} = \begin{cases} T_{s,a}^{\text{par}}, & \text{if } T_{s,a}^{\text{par}} < \theta \min_{a\in\mathcal{A}} T_{s,a}^{\text{par}}, \\ 0, & \text{otherwise,} \end{cases} \tag{84}$$

where θ is a coefficient of the minimum service data transmission time to remove the unacceptable transmission time. The feasible vertex label is satisfied: $w_s + w_a \leq w_{s,a}$, where $w_s = \min_{a\in\mathcal{A}} w_{s,a}$ and $w_a = 0$ are the vertex label of s and a in the KM algo-

rithm. Let $\mathbb{G}^{\text{mat}} = (\mathcal{S}, \mathcal{A}, < \mathcal{S}^{\text{mat}}, \mathcal{A}^{\text{mat}} >)$ be the equalling matching subgraph, satisfying $w_s + w_a = w_{s,a}$, where the link set $< \mathcal{S}^{\text{mat}}, \mathcal{A}^{\text{mat}} >$ is initialized to an empty set.

The perfect matching of the equalling matching subgraph $\mathbb{G}^{\text{mat}}$ can be denoted as M^*, and we have the following theorem.

Theorem 2. *Assuming w_s and w_a are the feasible vertex labels, if the equalling match subgraph $\mathbb{G}^{mat}$ has a perfect matching, M^*, M^* is also a perfect match with a minimum total weight of $\mathbb{G}$.*

Proof. The proof is analyzed in Appendix A. □

The steps of the KM algorithm can be elaborated as follows:

1. Initialize w_s, w_a, and $w_{s,a}$.
2. Enumerate $s \in \mathcal{S}$, find $a \in \mathcal{A}$ satisfies $w_s + w_a = w_{s,a}$ based on the Hungarian algorithm.
3. If $a \in (\mathcal{A} - \mathcal{A}^{\text{mat}}) \cap \mathcal{A}^{\text{rea}}$, add $< s, a >$ into $\mathbb{G}^{\text{mat}}$; otherwise, calculate the matching distance $z = \min_a\{w_{s,a} - w_s - w_a, s \in \mathcal{S}^{\text{rea}}, a \in (\mathcal{A} - \mathcal{A}^{\text{mat}})\}$, set $w_s = w_s + z, s \in \mathcal{S}^{\text{rea}}$ and $w_a = w_a - z, a \in \mathcal{A}^{\text{rea}}$. Then, change the reachable path into links, e.g., $s, < a_1 s^* >, a^*$ to $< s a_1 >, < s^* a^* >$.
4. Repeat 2 and 3 until obtaining M^* of $\mathbb{G}^{\text{mat}}$.

Therein, $\mathcal{S}^{\text{mat}}$ and $\mathcal{A}^{\text{mat}}$ are the vertex sets of two sides, whose elements have links in $\mathbb{G}^{\text{mat}}$, respectively. $\mathcal{S}^{\text{rea}}$ and $\mathcal{A}^{\text{rea}}$ are the searching reachable path sets of two sides by the breadth-first search in the Hungarian algorithm [29], respectively. s^* and a^* are the corresponding variables of z. The matching decision can be derived from:

$$x_{s,a}^{\text{mat}}(t) = \begin{cases} 1, & \text{if } < s, a > \in M^*, \\ 0, & \text{otherwise,} \end{cases} \tag{85}$$

After obtaining the result of the KM algorithm, a few individual MEC servers or mobile devices are allocated an unsatisfied cellular channel, which significantly delays the transmission time. We further design a worst-case arranging mechanism to deal with it:

$$x_{s,a}^{\text{mat}}(t) = \begin{cases} 0, & \text{if } w_{s,a} | < s, a > \in M^* = \max_{a \in \mathcal{A}} w_{s,a}, \\ 1, & \text{otherwise,} \end{cases} \tag{86}$$

if $x_{s,a}^{\text{mat}}(t) = 0$, s is scheduled to be allocated, a cellular channel in the next timeslot, which consumes less time when it is allocated a satisfied channel in the next timeslot.

The pseudo-code is shown in Algorithm 3. The complexity of the naive KM algorithm is $O(|\mathcal{S}|^4)$, and the KM algorithm with the slack array is $O(|\mathcal{S}|^3)$. The complexities of the allocated bandwidth divided method, link-initialized method, and worst-case arranging mechanism are $O(|\mathcal{M}(t)||\mathcal{N}(t)|)$, $O(1)$, and $O(1)$, respectively. Parts of cases 3 and 6 can be similarly solved in Algorithm 3.

Algorithm 3 KM algorithm-based channel-division fetching module (KCDF).

Require: total allocated bandwidth $b_0^{\text{par}}(t)$, responding set $\mathcal{S}$, cellular channel set $\mathcal{A}$, link threshold $\bar{T}_{s,a}^{\text{par}}$, transmission power $p_{s,a}(t)$, channel gain $h_{s,a}(t)$, co-channel interference $I_{s,a}(t)$

Ensure: matching decision $x_{s,a}^{\text{mat}}(t)$

1: Calculate $T_{s,a}^{\text{par}}$ based on the allocated bandwidth divided method according to (79).
2: Initialize $\mathbb{G}$ based on the link-initialized method according to (84).
3: Obtain the perfect matching M^* based on the KM algorithm.
4: Derive the matching decision $x_{s,a}^{\text{mat}}(t)$ according to (85).
5: Rearrange the matching decision $x_{s,a}^{\text{mat}}(t)$ based on the worst-case arranging mechanism according to (86).
6: **return** $x_{s,a}^{\text{mat}}(t)$.

5. Evaluation

5.1. System Implementation

For system implementation, we implement the framework in a real-world collaborative edge system testbed that consists of a Raspberry Pi4 Model B board (with 1.5 GHz CPU, 4 GB memory) and a desktop (with an Intel 8 Cores i7-10700F 2.90 GHz CPU and 16 GB memory). Raspberry Pi serves as the application server. The desktop serves as the MEC servers and mobile devices. All devices are connected under a local wireless router. We use the transmission control protocol (TCP) socket programming for guaranteeing reliable communication over all devices in the environment.

5.2. Case Study

We present a simulation of the proposed framework on the edge system testbed through a real-world image analysis case study: automatic license plate recognition. In particular, we leverage the convolutional neural network (CNN) framework as a service developed in [30]: an ImageNet model VGG-16. The VGG-16 model is a deep CNN with 16 layers for image recognition tasks and is trained in a distributed machine learning style. We use the open-source automatic license plate recognition dataset (available online: https://platerecognizer.com (accessed on 6 May 2022)) to emulate the tasks generated by mobile devices.

5.3. Experiment Setup

We use simulations to compare the performance of the framework. The hyperparameters of the simulation are as follows: the input size of the task is in $[2, 10]$ MB, the computation amount in [1000, 50,000] cycles, the MEC servers and mobile device number are in $\{10, 20, 30, 40\}$, the learning rate of the penalty function is in $\{0.1, 0.2, 0.3, 0.4\}$, the proportion of the training round is in $\{0.8, 0.85, 0.9, 0.95\}$, and the link-initialized coefficient is in $\{1.5, 1.8, 2.1, 2.4\}$.

Hence, we first choose some representative baselines compared with the LMKO module.

- **Fresh cache offloading priority (FCOP)**: An algorithm where the mobile device searches a MEC server with a fresh parameter cache and immediately offloads the task.
- **Cache offloading priority (COP)**: An algorithm where the mobile device searches a MEC server with cache and immediately offloads the task.
- **Offloading priority (OP)**: An algorithm where the mobile device searches a MEC server and immediately offloads the task.
- **Local execution with fresh cache priority (LEFC)**: An algorithm where the mobile device executes the task locally if it maintains a fresh parameter cache; otherwise, it offloads the task to a MEC server.

Moreover, we further pick competitive baselines compared with the LLUC module.

- **Queue backlog priority (QBP)**: An algorithm constrains the penalty weight in a relatively low range of the Lyapunov optimization.
- **Total bandwidth priority (TBP)**: An algorithm constrains the penalty weight in a relatively high range.
- **Queue backlog empty (QBE)**: An algorithm fixes the penalty weight to 0 of the Lyapunov optimization.
- **Fixed total bandwidth (FTB)**: An algorithm fixes the penalty weight in an extremely high value.

We also select a few representative strategies compared with the KCDF module.

- **Hungary algorithm (HA) [29]**: An algorithm is leveraged to solve the maximal matching problem of a non-weight bipartite graph.
- **Channel bandwidth allocated-based size (CBAS)**: An algorithm where the total bandwidth is allocated based on the responding service data size.

- **Channel bandwidth allocated-based case (CBAC)**: An algorithm where the total bandwidth is allocated based on the requesting offloading case.
- **Uniform allocation of channel bandwidth (UACB)**: An algorithm where the total bandwidth is allocated uniformly.

5.4. LLUC Evaluation

We first investigate the LLUC module to compare the performance of the time average–total bandwidth under different learning rates of the penalty weight. From Figure 2a, it can be shown that our proposed LLUC module with $\zeta = 0.1$ achieves the best result over the change of the requesting number. As ζ increases from 0.1 to 0.4, the performance degrades over all of the requesting numbers. A lower learning rate results in a relatively high penalty weight and the Lyapunov optimization minimizes the penalty. In the meantime, selecting a lower learning rate may lead to a higher backlog and further delay the response fetching time. Therefore, it is advisable to make a moderate selection to balance the bandwidth consumption and time overhead. Since using $\zeta = 0.2$ only increases bandwidth by 24.0% and decreases the queue backlog by 53.3% compared to using $\zeta = 0.1$ for 10 requests, we take $\zeta = 0.2$ as the learning rate, considering the bandwidth and time.

Then, we studied the LLUC module to compare the performance of the average AoI of each response service data under distinct round proportions of request rearrangements. In Figure 2b, the simulation results show that when $\eta = 0.8$, i.e., when a 20% interval is left until the next training, the algorithm consistently achieves the lowest AoI regardless of the number of requests. As η increases from 0.8 to 0.95, the average AoI becomes higher. More requests are rearranged to another timeslot for service data with lower AoI. However, the time latency can deteriorate while the response timeslots are delayed. Therefore, considering the responding transmission time and AoI service data, which decrease the service fetching frequencies, we selected a medium proportion $\eta = 0.9$ to balance the trade-off, whose average AoI only increases by 23.8% and the time latency decreases by 36.2%, when comparing $\eta = 0.8$ under 10 requests.

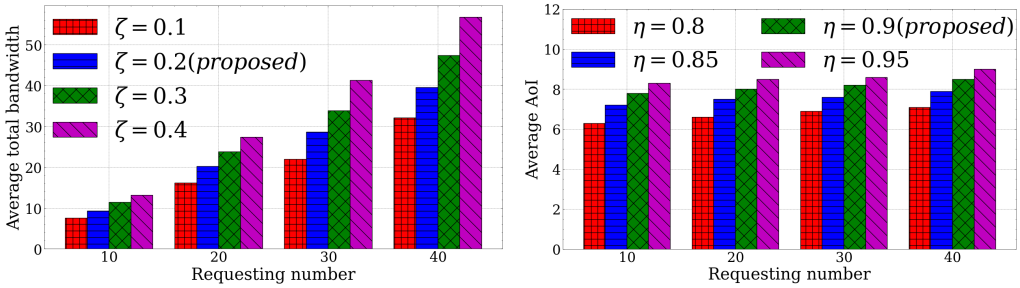
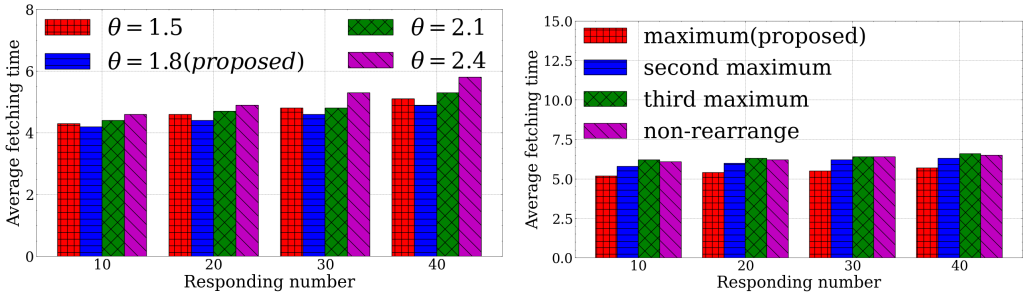


Figure 2. LLUC evaluation under different learning rates and training round proportions. (**a**) Learning rate. (**b**) Proportion of training round.

5.5. KCDF Evaluation

From the perspective of the module KCDF, we first evaluate the performance of the average fetching time under different link-initialized coefficients. Figure 3a plots that the KCDF with $\theta = 1.8$ outperforms other link-initialized coefficients as the responding number increases. It is concluded that a lower link-initialized coefficient can remove more unsatisfied links in the KM algorithms. As θ increases from 1.8 to 2.4, the average fetching time becomes higher. However, selecting a link-initialized coefficient that is too low such as $\theta = 1.5$ can increase the probability that the MEC server or mobile device fails to find a link in the equalling matching subgraph, which can significantly decline the performance. To make a feasible balance trade-off, we select $\theta = 1.8$ to guarantee the transmission time latency with at least a 2.3% performance improvement over the second-best result from $\theta = 1.5$ in 10 responses.



(**a**) (**b**)

Figure 3. KCDF evaluation under different link-initialized coefficients and rearrange conditions. (**a**) Link-initialized coefficient. (**b**) Rearrange condition.

Secondly, we compare the average fetching time under distinct rearrange conditions in Figure 2b. It is illustrated that rearranging based on the link weight is not less than the maximum, i.e., $w_{s,a}| < s, a > \in M^* = \max_{a \in \mathcal{A}} w_{s,a}$ has the minimum average fetching time when the responding number varies. The fetching time increases from the maximum, the second maximum, and the third maximum. When the application server rearranges based on the weight of the link is not less than the third, its result is even worse than the non-rearrangement. If the link is not the worst choice of the MEC server or mobile device, it is better to respond at this timeslot; otherwise, it suffers a higher fetching time. We choose the rearranging condition if the link weight is not less than the maximum to the LLUC module, which results in a time reduction of at least 3.7% compared to the second maximum case for under 10 responses.

5.6. Performance Comparison

5.6.1. Average Cost Comparison

From Figure 4a, we investigate the average cost of the distinct baselines of the LMKO modules. Our LMKO module achieves the minimum of Z_1 under different requesting numbers. The LMKO module is capable of obtaining the minimum cost by offloading the task to the best MEC server or local execution. The second-best result belongs to the FCOP algorithm since the mobile device chooses a MEC server with a fresh cache to offload. The performances of the COP and OP are poor owing to their extra service fetching times. The worst result is brought by the LEFC since it does not take advantage of offloading in the edge system. The LMKO module is efficient in terms of the cost of the time delay and energy consumption with at least a 4.1% performance gain compared to the second-best result from FCOP (for under 10 requests).

5.6.2. Average Total Bandwidth Comparison

Figure 4b illustrates the performance of the time-averaged total allocated bandwidth of the baselines of the LLUC baselines. The proposed LLUC module has a superior result comparing other baselines while the requesting number increases. The efficiency of the LLUC module maintains a controlled queue backlog while minimizing the total allocated bandwidth. In the meanwhile, the TBP algorithm obtains the second minimum result since it takes a higher penalty weight but causes a longer queue backlog. The QBP algorithm preferentially considers the queue backlog, leading to a medium result. The FTB algorithm delivers a high performance despite a fixed total bandwidth, due to its large backlog. The worst result belongs to the QBE algorithm, which keeps the 0 queue backlog, even as the bandwidth significantly increases. The LLUC module outperforms other baselines in regard to queue backlog stability and total allocation bandwidth with at least a 19.7% performance gain compared to the second-best result from TBP under 10 requests.

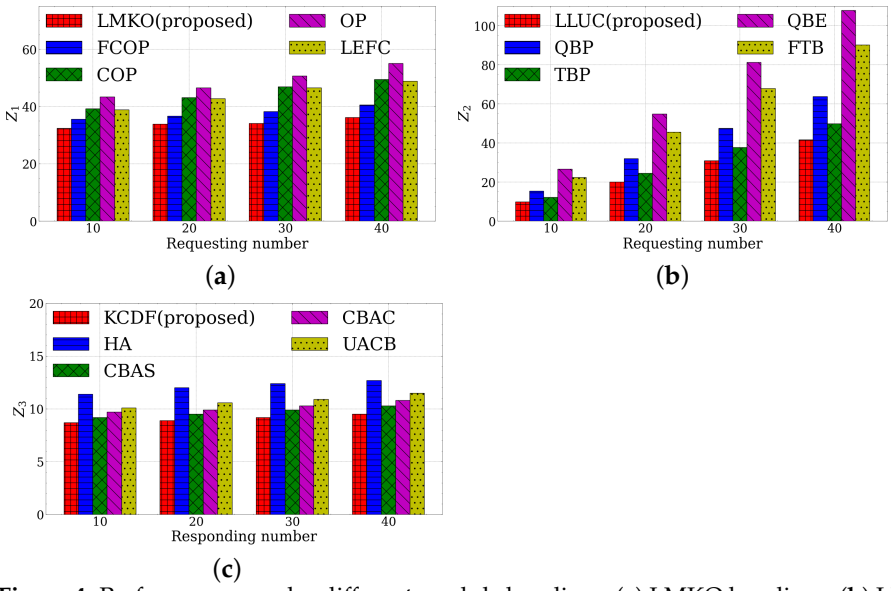


Figure 4. Performances under different module baselines. (**a**) LMKO baselines. (**b**) LLUC baselines. (**c**) KCDF baselines.

5.6.3. Average Fetching Time Comparison

Figure 4c displays the results of comparing the average fetching times of the baseline methods of the KCDF module. Our KCDF module exhibits superior performance across varying response numbers. The KCDF module finds a perfect match in the equalling matching subgraph, where each MEC server or mobile device matches its best-allocated cellular channel. The second lowest number belongs to the CBAS algorithm, which allocates the bandwidth according to the service data size. Each MEC server or mobile device can obtain the satisfied channel. The CBAC algorithm allocating the bandwidth with the offloading case suffers a similar situation with the CBAS and attains a medium result. The UACB has a poor result since it uniformly allocates the bandwidth leading to the response with service libraries and parameters having unsatisfied transmission latency. The HA algorithm suffers the worst result since it never updates the vertex label while it cannot find a link in the equalling matching subgraph, which results in a few vertices being matched, i.e., a few channels are allocated. Thus, our KCDF module has efficient performance in terms of average fetching time with at least a 2.2% performance gain compared to the second-best result from CBAS under 10 responses.

5.6.4. Average Time Cost of Baselines Combination

In Figure 5a, we investigated the performance of the average time cost of the baseline combinations, which includes our proposed ASCO framework (LMKO, LLUC, and KCDF modules) and other baselines achieving the competitive result consisting of FCOP, TBP, and CBAS algorithms. We can see that our ASCO framework always outperforms other baseline combinations while the time weight parameter changes, and the weights of energy and bandwidth remain. We minimize the average time cost by finding the most suitable offloading decision and allocating the best cellular channel. Other baseline combinations have declined results comparing our framework. LMKO+TBP+KCDF, LMKO+LLUC+CBAS, and LMKO+TBP+CBAS achieved moderate performance, as there was not a significant improvement in the modules. On the other hand, FCOP+LLUC+KCDF, FCOP+LLUC+CBAS, and FCOP+TBP+KCDF incurred higher costs, as their modules placed less emphasis on time concerns. Take FCOP+TBP+CBAS as an example, it had the worst performance due to the absence of the proposed modules. As ξ^{tim} increased, the performance gap between our framework and another baseline combination enlarged, which explains why the proposed framework has a significant gain in terms of time delay with at least a 9.6% improvement compared to the second-best result from LMKO+TBP+KCDF under $\xi^{\text{tim}} = 0.1$.

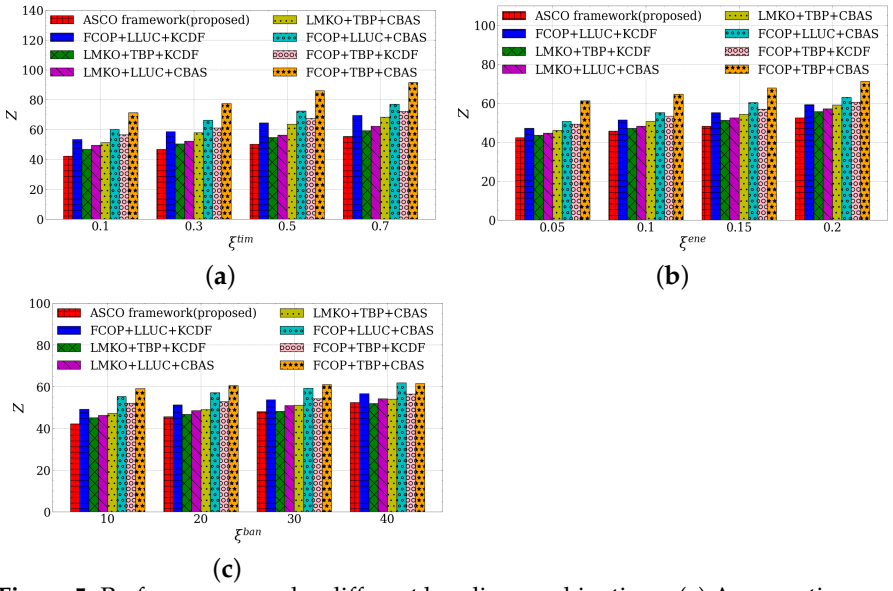


Figure 5. Performances under different baseline combinations. (**a**) Average time cost. (**b**) Average energy cost. (**c**) Average bandwidth consumption.

5.6.5. Average Energy Cost of Baselines Combination

Figure 5b shows the performance of the average energy cost of the combinations. Our ASCO framework keeps the best result while the energy weight varies and the weights of the time and bandwidth are fixed. The LMKO module makes an economized energy offloading decision to save the energy consumption of mobile devices. At the same time, other baseline combinations with the LMKO module outperform other combinations without the LMKO module due to the consideration of energy in the LMKO module. LMKO+TBP+KCDF, LMKO+LLUC+CBAS, and LMKO+TBP+CBAS obtain middle performances due to the lack of energy concerns. FCOP+LLUC+KCDF, FCOP+LLUC+CBAS, and FCOP+TBP+KCDF have higher costs since their modules are not efficient in terms of costs. Similarly, FCOP+TBP+CBAS had the worst performance. Hence, the average energy result verifies that our proposed framework achieves superior performance with respect to energy consumption with at least a 2.8% improvement compared to the second-best result from LMKO+TBP+KCDF under $\xi^{\text{ene}} = 0.05$.

5.6.6. Average Bandwidth Consumption of Baselines Combination

Figure 5c illustrates the comparison of the average bandwidth consumption allocated from the application server under the baseline combinations. The proposed ASCO framework attains minimum results except in an extreme case with $\xi^{\text{ban}} = 40$, while the weights of the time and energy are fixed. In this case, the emphasis on bandwidth allocation is extremely significant so that our framework only obtains the second-best performance while LMKO+TBP+KCDF has the best result. LMKO+LLUC+CBAS and LMKO+TBP+CBAS obtain middle performances because they do not well balance the total bandwidth and mobile device cost. FCOP+LLUC+KCDF, FCOP+LLUC+CBAS, FCOP+TBP+KCDF, and FCOP+TBP+CBAS always obtain the worse results due to the algorithm's inefficiency. However, the value of the bandwidth is impractical since it leads to relatively less consideration of the time delay and energy consumption. In most moderate-weight cases, our framework dominates other baseline combinations in regard to the average bandwidth consumption with at least a 6.2% improvement compared to the second-best result from LMKO+TBP+KCDF under $\xi^{\text{ban}} = 10$.

6. Conclusions

In our work, we consider a scenario of AoI-aware service caching-assisted offloading. The proposed ASCO framework consists of three modules: (1) the LMKO module based

on the method of Lagrange multipliers with KKT conditions. (2) The LLUC module based on the Lyapunov optimization. (3) The KCDF module based on the KM algorithm. The simulation results verify that the proposed ASCO framework outperforms other baseline combinations with respect to time overhead, energy consumption, and allocated bandwidth. The ASCO framework is efficient in the individual inference task and global bandwidth allocation and is viable to be practically deployed.

This work can be extended in several future directions. First, considering the proactive service caching, the MEC server can predict the offloading request and call for the application server for advanced fetching. Second, considering the task partition, if the tasks are partitioned before execution, the subtasks can be executed in distinct MEC servers or locally.

Author Contributions: Conceptualization, J.F.; Methodology, J.F.; Software, J.F.; Validation, J.F.; Formal analysis, J.F.; Investigation, J.F.; Resources, J.F.; Data curation, J.F.; Writing—original draft, J.F.; Writing—review & editing, J.G.; Visualization, J.F.; Supervision, J.G.; Project administration, J.F.; Funding acquisition, J.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62171481, National Key Research and Development Program of China grant number 2019YFE0114000, Special Support Program of Guangdong grant number 2019TQ05X150, Natural Science Foundation of Guangdong Province grant number 2021A1515011124 and the Science and Technology Program of Guangzhou under Grant 202201011577.

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.

Appendix A

Here is the proof of Theorem 2 from Section 4.3.

Proof. On the one hand, as $\mathbb{G}^{\text{mat}}$ is a generated equalling matching subgraph of $\mathbb{G}$, the perfect matching M^* of $\mathbb{G}^{\text{mat}}$ is also a perfect matching of $\mathbb{G}$, and $\sum_{<s,a>\in M^*} w_{s,a} = \sum_{s\in\mathcal{S}} w_s + \sum_{a\in\mathcal{A}} w_a$. On the other hand, any non-perfect matching M has $\sum_{<s,a>\in M} w_{s,a} \leq \sum_{s\in\mathcal{S}} w_s + \sum_{a\in\mathcal{A}} w_a$. Therefore, the perfect matching M^* has a minimum total weight equaling its upper bound. □

References

1. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
2. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
3. Feng, J.; Gong, J. Joint Detection and Computation Offloading With Age of Information in Mobile Edge Networks. *IEEE Trans. Netw. Sci. Eng.* **2022**, 1–14 . [CrossRef]
4. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
5. Parvez, I.; Rahmati, A.; Guvenc, I.; Sarwat, A.I.; Dai, H. A survey on low latency towards 5G: RAN, core network and caching solutions. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3098–3130. [CrossRef]
6. Wang, S.; Zhang, X.; Zhang, Y.; Wang, L.; Yang, J.; Wang, W. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access* **2017**, *5*, 6757–6779. [CrossRef]
7. Waqas, M.; Tu, S.; Halim, Z.; Rehman, S.U.; Abbas, G.; Abbas, Z.H. The role of artificial intelligence and machine learning in wireless networks security: Principle, practice and challenges. *Artif. Intell. Rev.* **2022**, *55*, 5215–5261. [CrossRef]
8. Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; Rellermeyer, J.S. A survey on distributed machine learning. *ACM Comput. Surv.* **2020**, *53*, 1–33. [CrossRef]
9. Kaul, S.; Yates, R.; Gruteser, M. Real-time status: How often should one update? In Proceedings of the 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 2731–2735.

10. Sun, Y.; Chen, Z.; Tao, M.; Liu, H. Bandwidth gain from mobile edge computing and caching in wireless multicast systems. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 3992–4007. [CrossRef]
11. Ning, Z.; Zhang, K.; Wang, X.; Guo, L.; Hu, X.; Huang, J.; Hu, B.; Kwok, R.Y. Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 2212–2225. [CrossRef]
12. Yang, X.; Fei, Z.; Zheng, J.; Zhang, N.; Anpalagan, A. Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing. *IEEE Trans. Veh. Technol.* **2019**, *68*, 11018–11030. [CrossRef]
13. Guo, H.; Liu, J. Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks. *IEEE Trans. Veh. Technol.* **2018**, *67*, 4514–4526. [CrossRef]
14. Wu, H.; Lyu, F.; Zhou, C.; Chen, J.; Wang, L.; Shen, X. Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 2783–2797. [CrossRef]
15. Ko, S.W.; Huang, K.; Kim, S.L.; Chae, H. Live prefetching for mobile computation offloading. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 3057–3071. [CrossRef]
16. Lyu, F.; Ren, J.; Cheng, N.; Yang, P.; Li, M.; Zhang, Y.; Shen, X.S. LEAD: Large-scale edge cache deployment based on spatio-temporal WiFi traffic statistics. *IEEE Trans. Mob. Comput.* **2020**, *20*, 2607–2623. [CrossRef]
17. Gu, Z.; Lu, H.; Zhu, Z. On throughput optimization and bound analysis in cache-enabled fiber-wireless networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9068–9082. [CrossRef]
18. Chen, L.; Xu, J.; Ren, S.; Zhou, P. Spatio–temporal edge service placement: A bandit learning approach. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 8388–8401. [CrossRef]
19. Xu, J.; Chen, L.; Zhou, P. Joint service caching and task offloading for mobile edge computing in dense networks. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 207–215.
20. Zhao, T.; Hou, I.H.; Wang, S.; Chan, K. Red/led: An asymptotically optimal and scalable online algorithm for service caching at the edge. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1857–1870. [CrossRef]
21. He, T.; Khamfroush, H.; Wang, S.; La Porta, T.; Stein, S. It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 365–375.
22. Ma, M.; Wong, V.W. Age of information driven cache content update scheduling for dynamic contents in heterogeneous networks. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 8427–8441. [CrossRef]
23. Xu, C.; Xie, Y.; Wang, X.; Yang, H.H.; Niyato, D.; Quek, T.Q. Optimal status update for caching enabled IoT networks: A dueling deep R-network approach. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 8438–8454. [CrossRef]
24. Zhang, S.; Li, J.; Luo, H.; Gao, J.; Zhao, L.; Shen, X.S. Low-latency and fresh content provision in information-centric vehicular networks. *IEEE Trans. Mob. Comput.* **2020**, *21*, 1723–1738. [CrossRef]
25. Boyd, S.; Boyd, S.P.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
26. Davis, D.; Drusvyatskiy, D.; Kakade, S.; Lee, J.D. Stochastic subgradient method converges on tame functions. *Found. Comput. Math.* **2020**, *20*, 119–154. [CrossRef]
27. Neely, M.J. Stochastic network optimization with application to communication and queueing systems. *Synth. Lect. Commun. Netw.* **2010**, *3*, 1–211.
28. Kuhn, H.W. Variants of the Hungarian method for assignment problems. *Nav. Res. Logist. Q.* **1956**, *3*, 253–258.[CrossRef]
29. Mills-Tettey, G.A.; Stentz, A.; Dias, M.B. *The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs*; Tech. Rep. CMU-RI-TR-07-27; Robotics Institute: Pittsburgh, PA, USA, 2007.
30. Li, H.; Wang, P.; Shen, C. Toward end-to-end car license plate detection and recognition with deep neural networks. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 1126–1136. [CrossRef]