Computation Offloading in Mobile Cloud Computing

March 2020

Guo Kai

Computation Offloading in Mobile Cloud Computing

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2020

Guo Kai

Abstract

With the growth of IoT technologies, mobile devices have become an indispensable part of daily life, as they provide data processing ability to mobile users anytime and anywhere. On one hand, daily life becomes immensely convenient according to executing various mobile applications on mobile devices. On the other hand, it is difficult for the limited resources of mobile devices, such as processing power, storage capacity and battery life, to satisfy the demands of current mobile applications, especially of computationally intensive applications. Thus, it is important to find a way to execute mobile applications while satisfying the demands on response time and energy consumption. Fortunately, a new paradigm, mobile cloud computing (MCC), is introduced to alleviate the problem of limited resources for mobile devices. Migrating (offloading) computations from mobile devices to servers in MCC paradigm can augment the computational capabilities and reduce the power consumptions for resource-scarce mobile devices. In order to get the highest offloading performance, it is important but difficult to find a proper strategy for computation offloading. In this thesis, we focus on the problem of how to offload computations from mobile devices to servers so as to minimize the response time of applications. Two offloading strategies are proposed to adapt different scenarios.

In chapter 3, we study the computation offloading problem in two-tier MCC system including mobile devices and an edge server. An application is considered as a set of dependent tasks each of which can be independently offloaded and the capacity limitations of both the communication channel and the edge server are taken into account. We firstly formulate an offline offloading problem as a mixed-integer linear programming problem. Then, we extend the offline problem to an online offloading problem in which an application can be executed at any time. Due to the complexity of the problem, it is difficult to obtain a solution within a realistic time period. Therefore, we propose an efficient approach based on congestion control. Extensive simulations demonstrate that our proposed approach significantly outperforms previous approaches.

In chapter 4, we study the offloading problem in three-tier MCC system including mobile devices, edge servers and cloud servers, in which the computations at mobile devices can be offloaded to edge servers or can be further offloaded to cloud servers if necessary. An application is considered to be an inseparable job that can be integrally offloaded and the capacity limitations of the communication channels, the edge servers and the mobile devices are taken into account. The problems of how to determine the offloading strategy and of how to determine the transmission rate of communication channels are jointly studied and formulated as an optimization problem. Then, we analyse the joint problem and transform it into a

piecewise convex optimization problem. An efficient algorithm that can find the optimal solution is proposed in our study. Extensive experiments show that our algorithm significantly outperforms previous algorithms. The experimental results also demonstrate high robustness of our algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we make an overview on computation offloading technology in mobile cloud computing from the origin of the mobile cloud computing paradigm to the detail of the computation offloading technology. We firstly discuss mobile computing and cloud computing, respectively. Then, we show the combination of cloud computing and mobile computing, i.e., mobile cloud computing paradigm. Thirdly, the computation offloading technology in mobile cloud computing paradigm is explained. After that, we introduce different kinds of computation offloading strategies and strategy-making processes. At last, the objectives and contributions of this thesis are shown.

1.1 Mobile Computing

Mobile computing involves mobile communication, mobile hardware, and mobile software, which allows data processing and transmission on mobile devices. With the development of IoT technology, mobile devices have become an indispensable part of daily life, as they provide data processing ability to mobile users anytime and anywhere. However, it is difficult for the limited resources of mobile devices, such as processing power, storage capacity and battery life, to satisfy the demands of computationally intensive applications. The power of a mobile device is usually limited due to its physical size so that the performance improvement brought by the hardware upgrade has been hard to meet the demands of people [1, 2]. Cloud computing technology has been proposed to alleviate the problem of limited resources for mobile devices.

1.2 Cloud Computing

Cloud computing is a style of easily scalable computing which is based on Internet and provides virtual resources as services over the Internet, such as computation, storage and applications [3]. These services can be rapidly provisioned and released in low cost, so that cloud computing can help users get the resources from the Internet easily without up-front infrastructure costs. The cloud computing paradigm is shown in Figure 1.1 and the advantages of cloud computing can be summarized as follows.

- Super-large scale. A cloud has a large number of servers, and can give users unprecedented computing power.
- Virtualization features. The services come from cloud are virtual. Thus, users can use these services anytime and anywhere.
- Generality. The same cloud can provide many different services and reply the different demands from different users.
- High scalability. A cloud can easily adjust servers to adapt to the different services.
- Cheapness. Users can use the cloud service without up-front infrastructure, and do not need to maintain and manage the cloud server.

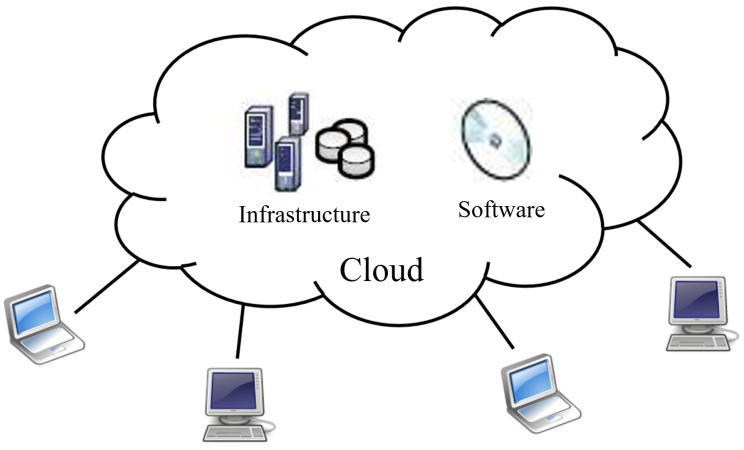


Figure 1.1. Cloud computing.

1.3 Mobile Cloud Computing

With the increase of mobile applications and the support of cloud computing for mobile users, mobile cloud computing was proposed as the extension of cloud computing in mobile scenarios which inherits the merits of the cloud computing and breaks through the limit of hardware. The mobile cloud computing paradigm as the combination of cloud computing and mobile computing extends the computation and storage capabilities of mobile devices and furthermore reduces the energy consumptions at mobile devices [4,5].

We show two types of mobile cloud computing network in Figure 1.2 and 1.3, respectively. In Figure 1.2, we show an example of mobile cloud computing network without cloud server. In this type, there is no cloud server in a mobile cloud computing system. Mobile devices are connected to each other via wireless channels [6]. An application executed at mobile device can be migrated to another device for saving energy consumption. However, it will increase total energy consumption of all of devices. For a device with relatively low computing power, executing its applications in other devices may reduce the response time. However, for a device with high computing power, executing its applications will delay the response time. In Figure 1.3, we show an example of mobile cloud computing network with a cloud server. In this type, there is a specialized cloud server for mobile cloud computing. Mobile devices connect to an access point (AP) via wireless channel, and the AP connects to a cloud server via high speed link. Because of the high processing speed of cloud server, it can be used for executing the mobile application remotely to reduce the response time of application.

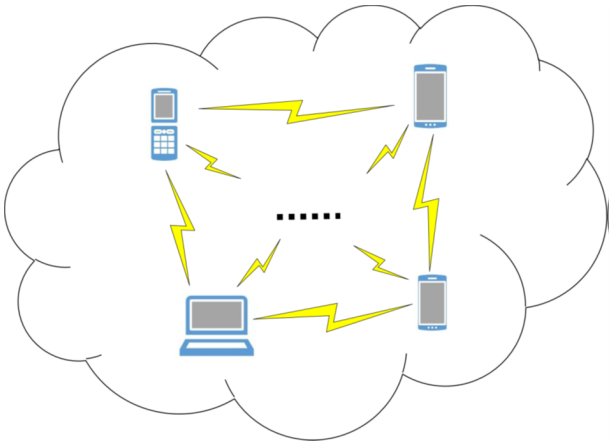


Figure 1.2. Mobile cloud computing (without cloud server).

Compared with the first type, the second type has lower total energy consumption and shorter application response time. Thus, we focus on the second type of

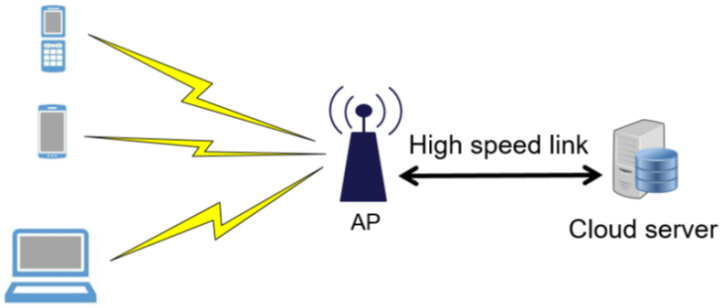


Figure 1.3. Mobile cloud computing (with a cloud server).

mobile cloud computing network in this thesis and study how to use the mobile cloud computing to reduce mobile application response time. However, since the remote cloud server is located on the Internet and far from mobile users in traditional mobile cloud computing paradigms, the communication delay between users and the cloud server is long, which degrades the performance of mobile cloud computing. A promising way is deploying a small-scale server, called an *edge server*, near by the AP as Figure 1.4 for reducing the communication delay in traditional mobile cloud computing [7,8]. The resources of the edge server will be preferentially provided for mobile users and that of the remote cloud server will also be provided for mobile users if necessary. How to efficiently use the resources of edge and cloud servers is a difficult but important problem in mobile cloud computing. In this thesis, we study how to use the resources of edge and cloud servers to reduce mobile application response time in mobile cloud computing.

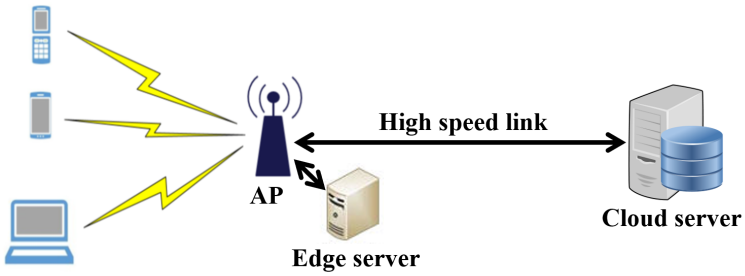


Figure 1.4. Mobile cloud computing (with an edge server).

1.4 Computation Offloading

Cloud computing paradigms can extend the end devices by offering "as a Service" frameworks. The core of cloud computing technologies is the the concept

of hardware virtualization – many virtual servers deployed in the same physical layer have independent operating environments to provide various services. Based on the specialities of services, cloud computing paradigms are presented in three cloud delivery models [9]: Infrastructure as a Service (IaaS) [10], Platform as a Service (PaaS) [11], and Software as a Service (SaaS) [12] as shown in Figure 1.5.

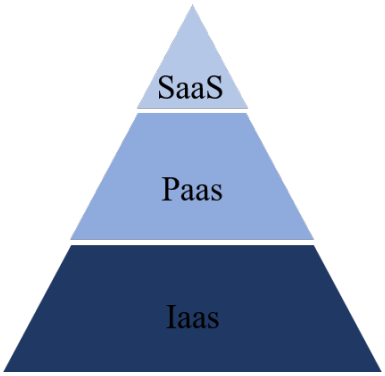


Figure 1.5. Cloud computing layers.

- IaaS. Only the infrastructures are provided for users. On the other word, users can have their own main frames which are located the Internet by IaaS. However, users have to manage the operating systems and software on the main frames.
- PaaS. The infrastructures and the operating systems are packaged as a platform and provided for users. It allows customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure.
- SaaS. Various software are provided for users under their demands. The software can be used without downloading and preinstallation. Certainly, in order to provide the software, services providers need to manage the infrastructures and the operating systems.

For users, IaaS has the highest freedom and SaaS has the highest convenience. For service providers, the infrastructures need to be managed no matter in IaaS, PaaS or SaaS. Thus, how to use the infrastructures effectively is extremely important.

The utilization of the infrastructures in mobile cloud computing is mainly divided into two categories: the utilization of storage capacity and the utilization of computing power. The former allows users to store their data on the servers and download them whenever and wherever. The latter allows users to offload their

computations from the mobile devices to the servers, which is also called *computation offloading*, in order to lighten the work of computing for mobile devices and reduce the response time of mobile applications [13,14].

In thesis, we focus on the computation offloading problem in mobile cloud computing and study how to effectively use the resources of servers to reduce the response time of mobile applications.

1.5 Application Level and Task Level Offloading Strategies

In order to speed up the computing of applications, a part of computation needs to be offloaded from mobile devices to servers. Offloading strategies can be mainly divided into two categories, i.e., application level offloading strategies and task level offloading strategies, which depend on the offloading granularity. Application level offloading is also called complete offloading. An application is considered to be an inseparable job that can be integrally offloaded from a mobile device to a server. Task level offloading is also called class level offloading, method level offloading, thread level offloading or partial offloading. An application is divided into a set of inseparable tasks each of which can be offloaded independently.

1.5.1 Application Level Offloading

We show an example of the application level offloading strategy in Figure 1.6. When an application is decided to be offloaded, the input data of the application will be transmitted to the server. Then, the server will executed the application and transmit the output data of the application back to the mobile device.

Since different applications are treated as independent parts, the performance of the system can be analysed and then find the optimal offloading strategies for application level offloading. However, the offloading problem is considered from a macro perspective in the application level offloading strategy, so that the optimal offloading performance of the application level offloading strategies may be lower than that of the task level offloading strategies.

1.5.2 Task Level Offloading

We show an example of an application which is divided into tasks in Figure 1.7, called as *task flow graph*. In Figure 1.7, there are 6 tasks, and the arrows mean execution sequence between two tasks. For example, task 1 cannot be executed

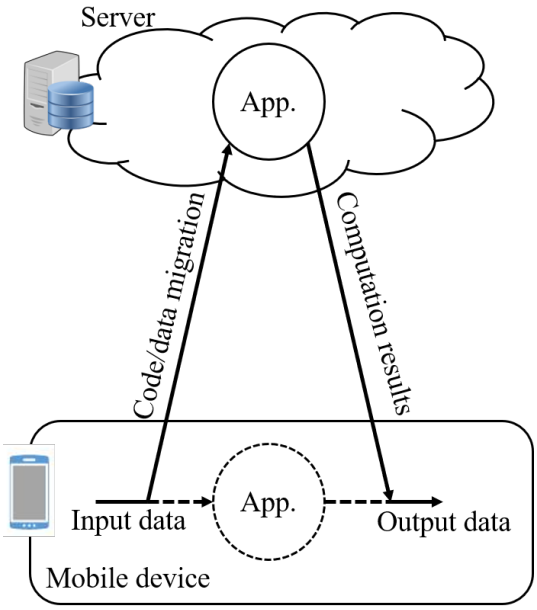


Figure 1.6. Application level offloading.

before the ending of execution of task 0, and task 3 cannot be executed before the ending of both tasks 1 and 2.

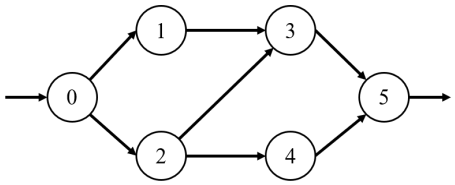


Figure 1.7. Task flow graph.

After the partition of application, we show an example of task level offloading in Figure 1.8. In this example, task 1 is decided to be offloaded to the server. After the execution of task 0, the input data of task 1 has been transmitted to the server. Then, task 1 is executed at the server and the output data will be transmitted back to the mobile device after the execution of task 1.

Since the offloading problem is considered from a micro perspective in the task level offloading strategy, the optimal offloading performance of task level offloading strategy may be better than that of the application level offloading strategies. However, an application is divided into a set of interrelated tasks, so that the optimal offloading strategy for task level offloading cannot be found in a short time.

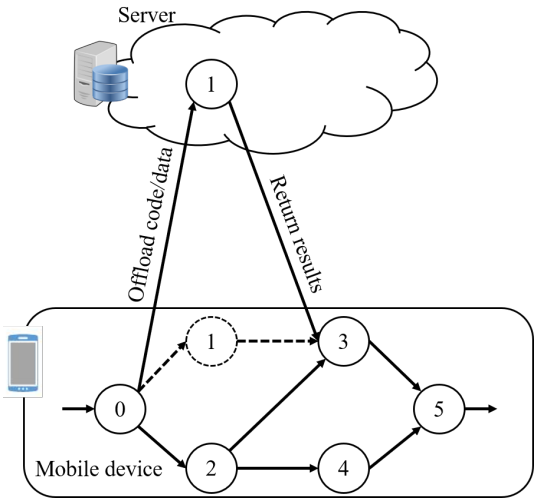


Figure 1.8. Task level offloading.

1.6 Offline and Online Offloading Approaches

In order to obtain the optimal offloading performance, we need to find the optimal offloading strategy. Strategy-making processes can be mainly divided into two categories, i.e., offline offloading approaches and online offloading approaches, which depend on the information of applications and system can be obtained in advance or not.

1.6.1 Offline Offloading

If the information of applications and the system including the arrival time and the workload of applications, the transmission rate of wireless communication channels and the computing power of mobile devices and servers can be obtained in advance, which is called *offline scenario*, an offline offloading approach can be used to determine the offloading strategy. An offline offloading approach determines the offloading strategy for all applications before the application arrivals such as Figure 1.9. When an application is arrived, the mobile device executes the application under the determined strategy.

It is possible to know the workload of an application, the transmission rate of wireless communication channels and the computing power of mobile devices and servers in advance. However, the arrival process of applications generally follows a stochastic process, and the offline approaches may become inefficient if we do not know the exact arrival time of each application in advance.

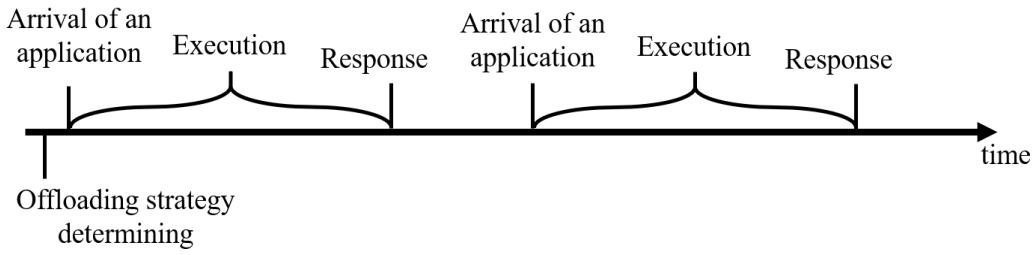


Figure 1.9. Offline offloading.

1.6.2 Online Offloading

If the information of applications and the system is unknown in advance, which is called *online scenario*, the offloading strategy is usually determined by an online offloading approach. The online approaches determine the offloading for each application after it arrives at the system such as Figure 1.10. When an application is newly executed, the information of the application are sent to the edge server, and the offloading strategy is determined and sent back to the mobile device. Then, the mobile device executes the application under the determined strategy.

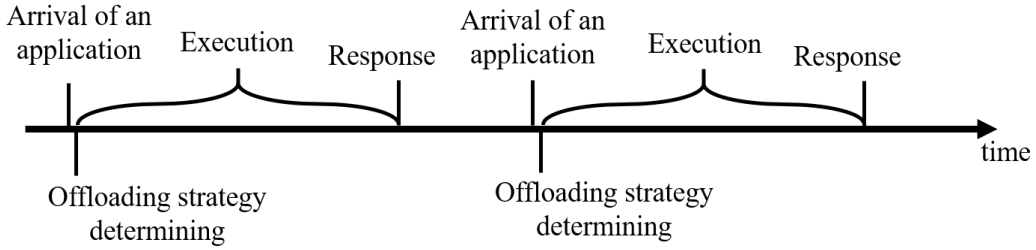


Figure 1.10. Online offloading.

In an online offloading approach, the offloading strategy of an application is determined when the applications is arrived, so that the application can be executed at any time. However, the extra latency, such as extra communication time and strategy determining time, may delay the response time of the application.

1.7 Objectives of This Thesis

In this thesis, we focus on the computation offloading problem in mobile cloud computing and aim at minimizing the average response time of applications. Two efficient offloading approaches in different offloading granularities are proposed and summarized as follows.

1. We propose an efficient heuristic task level online offloading approach. An application is divided into a set of inseparable and interrelated tasks which can be freely offloaded to an edge server via a wireless shared communication channel. Following this approach, the optimal offloading strategy of each task is found to minimize the average response time of applications for task level offloading. The main contents of this approach need to include: the construction of the edge computing model, the analysis of task flow graph, the assignment strategy of communication resources of the shared channel and computation resources of the edge server, and the scheduling of application execution.

2. We propose an application level offline offloading approach. An application is treated as an inseparable job and can be offloaded to either edge servers or remote cloud servers. Following this approach, the optimal application offloading ratio is found to minimize the average response time of applications for application level offloading. The main contents of this approach need to include: the construction of the cloud-assisted edge computing model, the analysis of the data transmission processes at the uplink channel and the downlink channel, the application execution processes at mobile devices and servers, and the concatenation of all these processes.

1.8 Contributions

Firstly, a task level online offloading approach is proposed to minimize the average response time of applications in mobile edge computing. We construct a mobile edge computing model where a set of mobile devices are connected with an edge server via a shared wireless communication channel. System characteristics such as the parallel processing of parallel tasks, the capacity limitation of the shared communication channel and the computation resource limitation of the edge server are considered. The main contributions of proposed approach can be summarized as follows.

1. We solve the offloading problem, in which the capacity limitation of the shared communication channel is considered and the edge server is no computation resource limitation. The problem can be solved by two proposed approaches, i.e., multi-user general-application (MUGA) algorithm and approach based on optimization technique (Opt. approach), respectively.

    - *MUGA.* We firstly solve the single-user chain-application (SUCA) problem. Then, the single-user general-application (SUGA) problem is solved

based on the solution of SUCA problem. At last, we consider the multi-user general-application (MUGA) problem which is solved based on the solution of SUGA problem.

- *Opt. approach.* We firstly formulate the offline offloading problem as the mixed-integer linear programming (MILP) problem [15] and extend the offline offloading problem to an online offloading problem for adaptation to online scenarios wherein an application can be executed at any time. Then, we use an appropriate software package such as Gurobi Optimizer [16] to solve the dynamic problem for each newly executed application in which the offloading decisions of the previously executed applications are not changed.

2. We solve the offloading problem, in which limitations of both the capacity of the shared channel and the computation resources of the edge server are considered. The initial solution is obtained by either the MUGA algorithm or the Opt. approach which depends on the degree of system congestion. Then, we adjust the obtained initial solution in order to meet the computation resources limitation of the edge server. As seen from simulation experiments, our proposed task level online offloading approach significantly outperforms previous task level offloading approaches.

Secondly, an application level offline offloading approach is proposed to minimize the average response time of applications. We construct a cloud-assisted edge computing model where a set of mobile devices are connected with an edge server wireless communication channels and the edge server is connected with a cloud server via a wired connection. The problems of both computation offloading and transmission rate determination are jointly studied. System characteristics such as the total bandwidth limitation of the communication channels and the computation resource limitation of the mobile devices and the edge server are considered. The main contributions of proposed approach can be summarized as follows.

1. We analyse the cloud-assisted edge computing model and formulate the joint problem as a multi-variable non-convex optimization problem in which the system characteristics are considered.

2. The formulated problem is transformed into a single-variable piecewise convex optimization problem based on our analysis.

3. An algorithm is proposed to efficiently find the optimal offloading ratios for each mobile decide and the edge server in order to minimize the average response of applications. Through experiments, we demonstrate that our proposed algorithm outperforms previous algorithms and the performance of our algorithm is highly robust.

1.9 Organization of This Thesis

The remainder of this thesis is organized as follows. In chapter 2, we review the related works on computation offloading. Chapter 3 presents the task level offloading problem in mobile edge computing. The corresponding model, optimization formulation, algorithms for task level offloading, performance evaluation and discussion are also given in this chapter. Then, Chapter 4 presents the application level offloading problem in cloud-assisted edge computing. The system model, optimization formulation, approach for application level offloading, performance evaluation and discussion are also given in chapter 4. Finally, chapter 5 concludes this thesis with a summary and gives the future work.

Chapter 2

Related Works

Many researchers have studied the offloading problem in mobile cloud computing. In this chapter, we firstly introduce some classical offloading frameworks to show the evolution of the studies on the computation offloading problem. Then, we summarize the offloading strategies proposed by previous works in order to explain the necessity of our works.

2.1 Various Offloading Frameworks

The studies of the computation offloading problem mainly focus on two issues: which computations need to be offloaded and where the computations need to be offloaded to. We show 5 offloading frameworks to explain the evolution of the consideration on these two issues.

2.1.1 Mirror Server

For a new technology, researchers tend to focus on how to implement the technology, rather than optimizing the details of the technology to obtain more benefits. Mirror Server is a classic example which is proposed in [17]. The authors focused on developing a framework that used remote services based on Telecommunication Service Provider to offload the computation from mobile devices to the server. For each mobile device, an unique mirror virtual-machine (VM) is created on the mirror server. An application is integrally offloaded and remotely executed in the mirror VM. After the remote execution, mirror server will return the result to the mobile device.

The Mirror Server architecture is presented in Figure 2.1. On each mobile device, a client synchronization module (Syn-client) is deployed within the device operating system (OS) to collect smartphone input data and transmit them to the

mirror server for synchronization. On the mirror server, in order to keep mirrors and mobile devices synchronized, the server synchronization module (Syn-server) updates mirrors using the data provided by Syn-client and the traffic monitor module which is deployed for collecting the network traffic between mobile devices and the IP network.

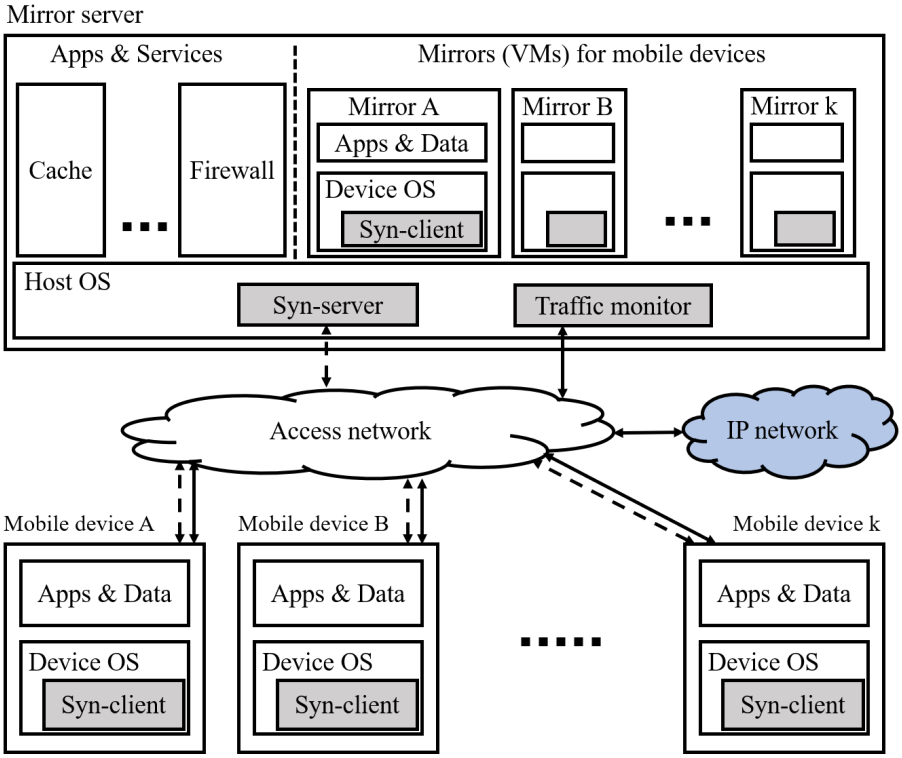


Figure 2.1. Mirror Server architecture (adapted from [17]).

This work showed the advantages of computation offloading as follows.

- The energy consumption of mobile devices can be saved.
- The execution of applications on the mirror will be much faster than on the mobile device because the mirror is much powerful than the mobile device.

Such work validated the feasibility of computation offloading and laid a solid foundation for later studies on computation offloading. In this thesis, we study how to efficiently implement the computation offloading in mobile cloud computing.

2.1.2 Cuckoo

After the implementation of computation offloading technology, some researchers considered the problem of that how to selectively offload part of applications or the part of an application under an offloading strategy instead of blindly offloading all applications from mobile devices to servers. Cuckoo as a classic framework was proposed in [18] and focus on selectively offloading a part of an application.

Cuckoo framework offloads an application to the cloud server using the Java stub/proxy model. This framework pre-partitioned an application into computation intensive and non-intensive methods using the existing activity model in Android when the application was developed. The computation intensive methods can be offloaded to the cloud server. When the application was executed, Cuckoo framework would check the networking environment and offload computation intensive methods to the cloud server if remote resource was reachable.

The main contribution of their work can be summarized as follows.

- They revealed that the computations can be selectively offloaded. In Cuckoo framework, a partial offloading strategy was determined.
- In order to partially offload an application, a pre-partition method was realized in Cuckoo framework.

Although the offloading strategy determinations in the works such as Cuckoo framework are very simple, such works enabled the selective offloading and promoted the maturity of the offloading strategy determination. In this thesis, we attempt to find efficient offloading strategies in order to minimize the average response time of applications.

2.1.3 MAUI

With the emergence of offloading strategy determination, some researchers focused on the problem of that how to determine an effective offloading strategy to minimize the energy consumption of a mobile device and the response time of an application. MAUI framework was proposed in [19] to minimize the energy consumption of the mobile device.

The MAUI architecture is presented in Figure 2.2. In the preparation step, the application was divided into a set of dependent methods and the code of the application must be installed into both mobile device and MAUI server. The profiler on mobile device measured and updated the characteristics of the device and the application. The profiler on MAUI server kept monitoring characteristics of the server and the network. Once an application was executed, MAUI server had to found an offloading strategy and send back to mobile device. Then, the application

was executed under the determined offloading strategy. In order to find the optimal offloading strategy, the offloading problem was formulated as an ILP problem and solved by a solver on the MAUI server. Client proxy and server proxy were responsible for control and data transmission between the mobile device and the MAUI server. MAUI controller was responsible for authentication and control of execution.

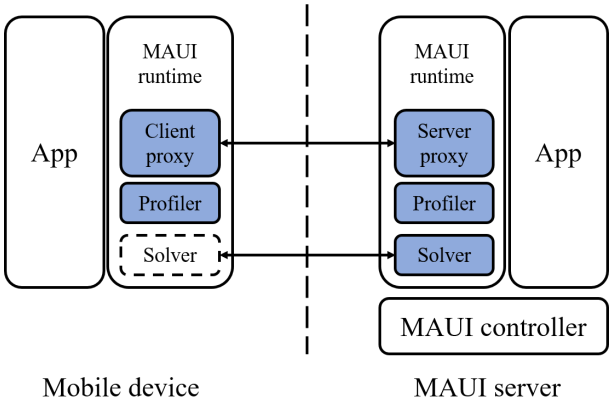


Figure 2.2. High-level view of MAUI architecture (adapted from [19]).

The main contribution of their work can be summarized as follows.

- A specific object, i.e., the optimization of energy consumption, was set for computation offloading and an online determining mechanism was proposed to achieve the object.
- A real-time feedback mechanism was proposed to enable the online determining.

Although the authors in [19] only focused on the computation offloading problem for few simple applications in a single-user system, the mechanism they proposed enabled the computation offloading in the online scenarios. In this thesis, we study the offloading problem and propose offloading strategies which can be used in the online scenarios.

2.1.4 Cloudlet

At the same time, there were also some researchers focused on an other problem of where the computations can be offloaded to. It is not always a wise way that offloading applications to the cloud server because of the high WAN latencies, especially for latency-intolerant applications. Authors in [20] suggested a promising

way of moving the server from remote Internet to the vicinity of the mobile device. They proposed the vm-based cloudlet framework to reduce the communication delay in computation offloading. The server was deployed in single-hop nearness to mobile devices and connected with mobile devices via wireless communication channels.

The Cloudlet illustration is shown in Figure 2.3. The environment of the mobile device was cloned and stored at the cloudlet server. When a mobile device executed, the whole application was offloaded to the server. After the execution, output of the application was transmitted back to the mobile device. The mobile device only served as a thin client to provide the user interface.

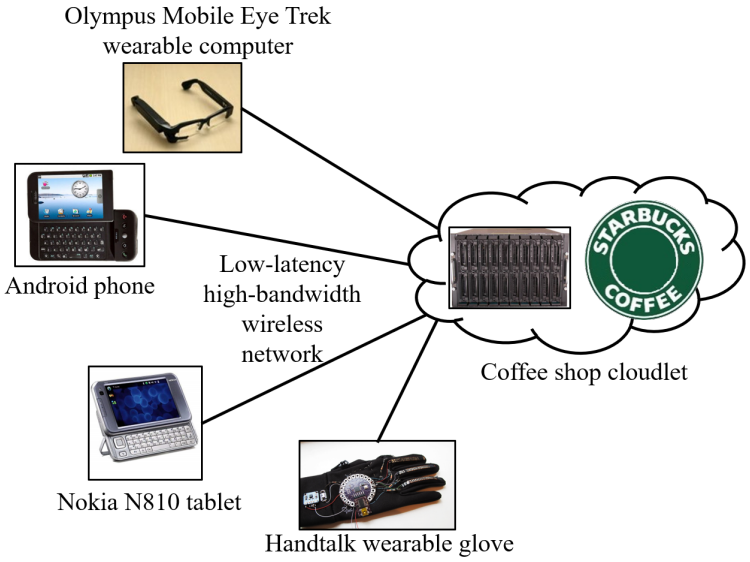


Figure 2.3. Cloudlet illustration (adapted from [20]).

The main contribution of their work can be summarized as follows.

- Cloudlet framework is proposed and deployed in this work to offload the computation from mobile devices with low latencies.

Such work provided a new way for computation offloading and evolved into the mobile edge computing paradigm in following studies. In chapter 3, we study the problem of how to efficiently offload tasks of applications from mobile devices to the edge server.

2.1.5 MAPCloud

Since the computing power of the edge server is limited, there were also some researchers considered that the computations can be offloaded to not only the edge

server but also the remote cloud server. A three-tier cloud computing framework, called MAPCloud, was suggested in [21–23]. In MAPCloud framework, the computations of an application was decided to be executed at the mobile device or offloaded to either the edge server or the remote cloud server in order to maximize the benefit of offloading.

The MAPCloud illustration is presented in Figure 2.4. An application was considered as a simple task-chain with a set of serial tasks. The benefit of offloading depended on where the tasks were executed and the offloading problem was formulated as an optimization problem. Since formulated problem is NP-hard, a heuristic algorithm was proposed to solve this problem.

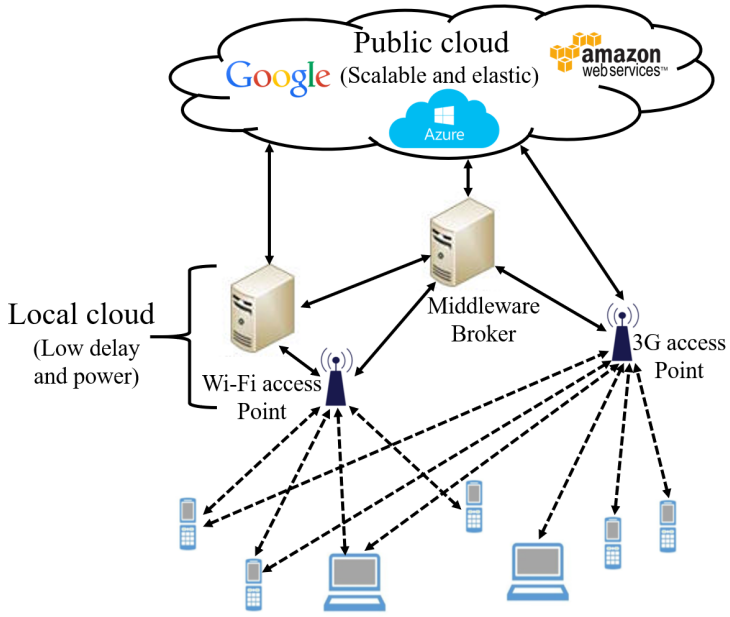


Figure 2.4. MAPCloud illustration (adapted from [21]).

The main contribution can be summarized as follows.

- The advantages of computation offloading in three-tier cloud computing was shown in these works. Comparing to the public cloud, three-tier cloud computing decreases energy consumption and execution delay. Comparing to the local cloud, they showed that three-tier cloud computing increases the scalability.

These works provided a new consideration of the joint utilization of the edge server and the remote cloud server in mobile cloud computing, which has become a hot issue in current studies on computation offloading [24]. In chapter 4, we study the problem of how to efficiently offload applications from mobile devices to either the edge server or the cloud server.

2.2 Various Offloading Strategies

Various offloading strategies for the computation offloading problem were proposed in previous works. These works are classified depending on the offloading granularity and summarized as follows.

2.2.1 Previous Application Level Offloading Strategies

Authors in [25–35] focused on the application level offloading problem. These studies can be mainly divided into two categories: one analysed the mobile cloud computing system from the viewpoint of game theory as [25–31] and the other one analysed the system from the viewpoint of queuing theory as [32–36]. The former formulated the offloading problem as a competitive game and found the Nash equilibrium in their formulated problem. Each mobile device was treated as a competitor that considered only its gain.Thus, fairness between users was fully considered in this kind of studies. However, the total gain of all of the mobile devices may not be appropriate when the Nash equilibrium is achieved. Moreover, they considered that all of the applications were arrived in the same time, which is almost impossible in a real system. To deal with the offloading problem in which an application can be executed at any time, the latter treated application executions and data transmissions as queues and analysed the offloading problem based on queuing theory. Offline determination approaches in their works that can adapt to the online scenario. However, authors in [32,33] did not consider the limitation of the communication channels, and authors in [34] did not consider the limitation of the server. Authors in [35] considered the limitations of both of the communication channels and the server but offloaded all applications from mobile devices to servers, which may cause low scalability of their approach. Authors in [36] attempted to minimize the sum of computation and communication cost where the computation on a mobile device can be vertically offloaded to other mobile devices and horizontally offloaded to edge servers and cloud servers. However, they considered that there an unique communication channel between a mobile device and an edge server, which may cause low utilization of total communication resource.

Thus, in this thesis, we consider the application level offloading problem from the viewpoint of queuing theory in which each application can be freely offloaded from a mobile device to a server and propose an offline offloading determination approach to find the optimal offloading strategy in chapter 4. Different with [32–35], we account for all of the resource limitations on the mobile devices, the communications channels and the edge server. Additionally, different with [36], we consider that the communication resource can be shared by multiple mobile devices.

2.2.2 Previous Task Level Offloading Strategies

Authors in [37–49] considered the task level offloading problem, in which each application is segmented into a number of dependent or independent tasks. Authors in [37–45] considered the offloading problem in a single-user mobile cloud computing system. Authors in [37–40] focused on reducing the application response time. Authors in [41] tried to jointly reduce both the energy consumption and application response time. Authors in [42–45] attempted to reduce the energy consumption of an application. Authors in [37] attempted to find the optimal offloading strategy before executing each application on a mobile device. Authors in [38] proposed a heuristic offloading algorithm for applications with serial tasks to reduce the application response time. Authors in [39] considered that an application consists of purely parallel or purely serial tasks and proposed a heuristic offloading algorithm to balance the load between the mobile and the server. Authors in [40, 42] considered both parallel and serial tasks in the same application and attempted to find an offloading strategy for reducing either the application response time or the energy consumption. The parallel tasks were considered to be processed serially in their works. Authors in [41, 43] considered multiple server sites for offloading the computation from mobile devices. Authors in [46–49] considered the offloading problem in a multi-user mobile cloud computing system. Authors in [46, 47] segmented an application into a set of independent tasks and tried to reduce the application response time in which the limitations of the shared channel and the server were considered. Authors in [48, 49] focused on the serial tasks offloading problem and aimed at reducing the application response time. The former considered the limitation of the server and the latter considered the limitations of both of the shared channel and the server.

All of above studies made their contribution for task level offloading problem from various aspects. However, non of them focused on the general application, i.e., consisting of parallel and serial tasks, task level offloading problem in an online scenarios while jointly considered the limitations of both the shared channel and the server. Thus, we study it in chapter 3 and aim at finding an offloading strategy to reduce the application response time.

Chapter 3

Task Level Online Offloading Strategy

In this chapter, we study the task level offloading problem in online scenarios while jointly considering the limitations of the resources of the shared channel and the server, and propose a task level online offloading strategy to minimize the average response time of applications. In this chapter, we firstly show the system model and formulate the offline offloading problem as a MILP problem. Then, we extend the offline problem to an online problem and formulate the problem for the case in which the execution times of applications are not known in advance. Furthermore, we propose a set of algorithms to solve the online offloading problem. After that, the results of simulation experiments are shown for examining our proposed algorithms. At last, we present the conclusions of our study on task level offloading problem. The findings in this chapter has been published in [50–52].

3.1 System Model

3.1.1 Edge Computing System

In this study, we consider the edge computing system shown in Figure 3.1. A set of mobile devices, denoted by $\mathcal{N} = \{1, 2, ..., N\}$, are connected to an edge server that is deployed in the access network near the mobile devices and serves as a small-scale cloud server for the mobile devices. The mobile devices are connected to an AP via a shared wireless communication channel and the AP is connected to the edge server via a high-speed link. Because the edge server is near by the AP, the transmission delay between the AP and the edge server can be neglected. We assume that there is only one application at most that is executed on a mobile device at a time. Some current mobile devices are equipped with muticore CPUs,

such as the Qualcomm Snapdragon 835 (8-core CPU) [53] and the MediaTek Helio X20 (10-core CPU) [54]. Thus, in this study, we assume that each mobile device has sufficient CPU cores to execute an application, meaning that no task needs to wait for execution once all of its input data are ready. To clearly describe the system, we denote the core of a mobile device that can process multiple tasks simultaneously as the core 0, and its processing rate is denoted by f_0. We assume that the server has a set of homogeneous CPU cores denoted by $\{1, 2, ..., L\}$, and their processing rates are denoted by $\{f_1, f_2, ..., f_L\}$. Furthermore, all the cores of the mobile devices and server are denoted as $\mathcal{L} = \{0, 1, 2, ..., L\}$.

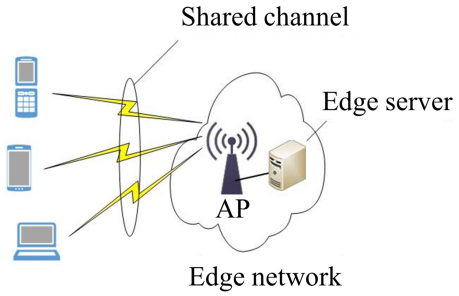


Figure 3.1. Edge computing system.

The network model considered in this study is shown in Figure 3.2. In this model, a set of mobile devices are connected to the edge server via a shared communication channel. Because of the shared channel, mobile devices may compete with each other for data transmission if they transmit data simultaneously. We assume that an ongoing data transmission cannot be interrupted by any other transmission, and we denote the transmission rate as s.

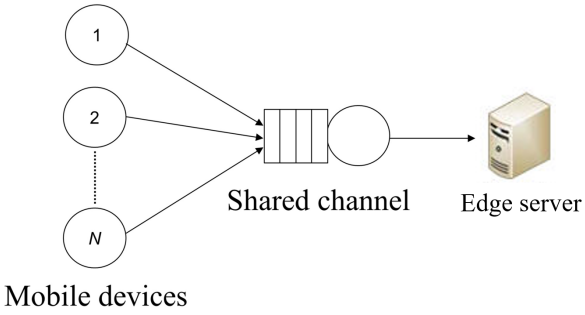


Figure 3.2. Network model.

3.1.2 Mobile Application

The general application executed at mobile device $i \in \mathcal{N}$, denoted simply as application i, or user i, mobile device i, constitutes a set of parallel and serial tasks, denoted as $\mathcal{M}^i = \{0, 1, 2, ..., M^i\}$, as shown in Figure 3.3, and the computation workload of task j is denoted as C_j^i. If necessary, a task computation can be interrupted. Only the tasks 0 and M^i of application i are un-offloadable, which means that the other tasks can be offloaded to the edge server for remote computing. We represent the relationship between tasks by a directed task flow graph as Figure 3.3. The directed arcs between nodes, denoted by $\mathcal{E}^i$, indicate data flows between tasks, and the directed arc from task j to task k, denoted by $e_{j,k}^i \in \mathcal{E}^i$, indicates that the output data of task j is sent to task k. The quantity of data transmitted from task j to k of application i is denoted as $d_{j,k}^i$, and the transmission delay of this data over the shared channel is $d_{j,k}^i/s$. However, if the two connected tasks j and k of the application i are calculated on the same side, i.e., at either the mobile device or the edge server, we assume that the delay of the data transmission from j to k can be ignored. We assume that the outputs of a task to its subsequent tasks are generated simultaneously and that these tasks, such as task 1 and task 2 shown in Figure 3.3, can be processed in parallel if there are enough CPU cores at the server. We also assume that a task can be computed only if all of the input data from its previous tasks have been received. For example, task 4 can only be computed after the output data from tasks 1 and 2 are received.

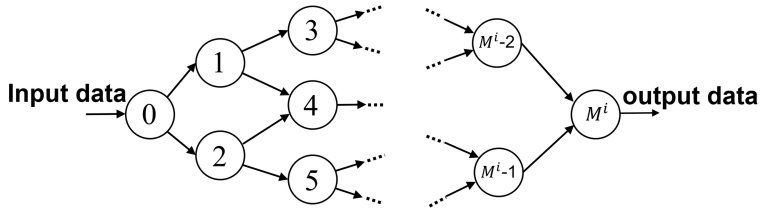


Figure 3.3. Application model.

We also assume that a task computation can be interrupted and that the core computing the task at the edge server can be switched to another if necessary. For example, as shown in Figure 3.4, the computation of a task can be divided into two segments that are processed by two different cores.

In this study, we focus on how to offload the tasks of mobile applications such that the average response of applications is minimized. The response time of an application is the time elapsed from the instant when the application is executed on a mobile device to the instant when the mobile device receives the execution result of the application. Information of an application, such as the relationship between

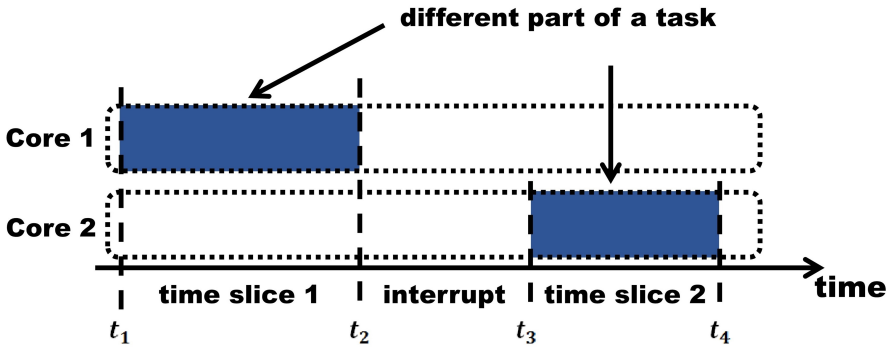


Figure 3.4. Task computation at the edge server.

tasks, the size of data transmitted between tasks and the computation workload of each task, can be obtained by the mobile device and sent to the server. The code of each application is pre-installed at both of the mobile device and the edge server. The server also has the knowledges of the transmission rate of the shared channel and the processing rates of the cores on the mobile devices and the server. Thus, the server can decide which tasks should be offloaded and send the offloading strategy back to the mobile device. This offloading strategy includes the offloading decision of each task, the task computing schedule and the data transmission schedule. Then, the mobile device can execute the application under the decided offloading strategy. Since the data size of the relevant information of an application and its offloading strategy is negligible compared with the size of data sent between tasks, we only consider the delays of data transmissions between tasks in this study.

3.2 Offline Offloading Problem

In offline offloading problem, the information of all application is sent to the server in advance and the offloading strategy is decided before the start of application executions.

A set of variables $\{x_j^{i,0}, x_j^{i,1}, ..., x_j^{i,L}\}$ are used to represent the proportions that how much of task j of application i is computed, i.e., $x_j^{i,l}(l \in \mathcal{L})$ denotes the proportion of task j of application i is processed by core l. Since the mobile device has enough CPU cores as described above, a task is computed either the whole task or none of the task on the mobile device, i.e., $x_j^{i,0} \in \{0,1\}$. Inversely, for the server shared by multiple users, we have $x_j^{i,l} \in [0,1](l \in \mathcal{L} \setminus \{0\})$. We assume that for a task, the processing cannot be split between the mobile device and the server. The beginning and the ending times of the processing of the partial workload of

task j of application i processed by core l are denoted as $\tau_j^{i,l}$ and $T_j^{i,l}$, respectively. The beginning and ending times of task j of application i is denoted as τ_j^i and T_j^i, respectively, where $\tau_j^i = min\{\tau_j^{i,l}\}(l \in \mathcal{L})$ and $T_j^i = max\{T_j^{i,l}\}(l \in \mathcal{L})$. For example in Figure 3.4, task j of application i is divided into two segments that are separately processed by cores 1 and 2 of the server, and we have $\tau_j^{i,1} = t_1$, $T_j^{i,1} = t_2$, $\tau_j^{i,2} = t_3$, $T_j^{i,2} = t_4$, $\tau_j^i = t_1$, and $T_j^i = t_4$. An integer variable $y_{j,k}^i$ is used to represent the whether two tasks j and k of application i are executed on the same side. $y_{j,k}^i = 0$ if tasks j and k are executed on the same side, i.e., either on the mobile device or the server, and $y_{j,k}^i = 1$ otherwise. $\omega_{j,k}^i$ and $W_{j,k}^i$ represent the beginning and ending times of the data transmission from task j to k, respectively. The beginning time of the execution of application i is denoted as δ^i. The notation used in this study is summarized in Table 3.1, 3.2 and 3.3.

Table 3.1. Parameters used in task level offloading problem.

Symbol	Meaning
$\mathcal{N}$	set of all applications in the system
$\mathcal{L}$	set of cores, $\mathcal{L} = \{0, 1, ..., L\}$
$\mathcal{M}^i$	set of tasks of application i
M^i	number of tasks of application i, $M^i = \left\|\mathcal{M}^i\right\|$
$\mathcal{E}^i$	set of directed links connecting two tasks of application i
$e_{j,k}^i$	link from task k to task j in the task flow graph of application i
C_j^i	workload of task j of application i
$d_{j,k}^i$	data size of the transmission from task j to task k of application i
f_l	processing rate of core l
s	channel transmission rate
I	a positive integer constant that is larger than any value we consider for a variable in this study
δ^i	beginning time of the execution of application i

Then, the task level offline offloading problem can be formulated as the following optimization problem.

$$\min \qquad T = \frac{1}{N} \sum_{i \in \mathcal{N}} \left(T_{M^i}^{i,0} - \tau_0^{i,0}\right), \tag{3.1}$$

subject to

$$\sum_{l \in \mathcal{L}} x_j^{i,l} = 1, \quad j \in \mathcal{M}^i, i \in \mathcal{M}, \tag{3.2}$$

Table 3.2. Decision variables used in task level offloading problem.

Symbol	Meaning
$x_j^{i,l}$	representing the proportion of the workload of task j of application i that is processed by core l, $x_j^{i,0} \in \{0,1\}$, $x_j^{i,l}(l>0) \in [0,1]$
$\tau_j^{i,l}$	representing the beginning time of the processing of a segment of task j of application i on core l
$\omega_{j,k}^{i}$	representing the beginning time of data transmission from task j to task k of application i

Table 3.3. Intermediate variables used in task level offloading problem.

Symbol	Meaning
τ_j^i	beginning time of the computation of task j of application i, $\tau_j^i = \min\{\tau_j^{i,l}\}(l \in \mathcal{L})$
T_j^i	ending time of the computation of task j of application i, $T_j^i = \max\{T_j^{i,l}\}(l \in \mathcal{L})$
$T_j^{i,l}$	ending time of the processing of a segment of task j of application i on core l
$W_{j,k}^i$	ending time of data transmission from task j to task k of application i
$y_{j,k}^i$	a binary integer variable indicating whether two consecutive tasks j and k of application i are executed on different sides, i.e., either on the mobile device or on the server
$z_j^{i,l}$	a binary integer variable indicating whether a segment of task j of application i is processed by core l
$\alpha_j^{i,l,l'}$	a binary integer variable indicating whether the processing of a segment of task j of application i on core l precedes the processing of another segment of this task on core l'
$\beta_{j,j'}^{i,i',l}$	a binary integer variable indicating whether the processing of a segment of task j of application i on core l precedes the processing of a segment of another task, i.e., task j' of application i', on core l
$\gamma_{j,k,j',k'}^{i,i'}$	a binary integer variable indicating whether the data transmission from task j to task k of application i precedes another data transmission from task j' to task k' of application i'

$$T_j^{i,l} - \tau_j^{i,l} = x_j^{i,l} C_j^i / f_l, \quad j \in \mathcal{M}^i, i \in \mathcal{N}, l \in \mathcal{L}, \tag{3.3}$$

$$\left|T_j^{i,l} - T_j^{i,l'}\right| + \left|\tau_j^{i,l} - \tau_j^{i,l'}\right| \geq x_j^{i,l} C_j^i / f_l + x_j^{i,l'} C_j^i / f_{l'},$$
$$j \in \mathcal{M}^i, i \in \mathcal{N}, l, l' \in \mathcal{L}, l \neq l', \tag{3.4}$$
$$\left|T_j^{i,l} - T_{j'}^{i',l}\right| + \left|\tau_j^{i,l} - \tau_{j'}^{i',l}\right| \geq x_j^{i,l} C_j^i / f_l + x_{j'}^{i',l} C_{j'}^{i'} / f_l,$$
$$j \in \mathcal{M}^i, j' \in \mathcal{M}^{i'}, i, i' \in \mathcal{N}, (i,j) \neq (i',j'), l \in \mathcal{L} \setminus \{0\}, \tag{3.5}$$
$$y_{j,k}^i = \left|x_j^{i,0} - x_k^{i,0}\right|, \quad j,k \in \mathcal{M}^i, e_{j,k}^i \in \mathcal{E}^i, i \in \mathcal{N}, \tag{3.6}$$
$$W_{j,k}^i - \omega_{j,k}^i = y_{j,k}^i d_{j,k}^i / s, \quad j,k \in \mathcal{M}^i, e_{j,k}^i \in \mathcal{E}^i, i \in \mathcal{N}, \tag{3.7}$$
$$\left|W_{j,k}^i - W_{j',k'}^{i'}\right| + \left|\omega_{j,k}^i - \omega_{j',k'}^{i'}\right| \geq y_{j,k}^i d_{j,k}^i / s + y_{j',k'}^{i'} d_{j',k'}^{i'} / s, \quad j,k \in \mathcal{M}^i,$$
$$j',k' \in \mathcal{M}^{i'}, e_{j,k}^i \in \mathcal{E}^i, e_{j',k'}^{i'} \in \mathcal{E}^{i'}, i, i' \in \mathcal{N}, (i,j,k) \neq (i',j',k'), \tag{3.8}$$
$$\omega_{j,k}^i \geq T_j^{i,l}, \quad j,k \in \mathcal{M}^i, e_{j,k}^i \in \mathcal{E}^i, i \in \mathcal{N}, l \in \mathcal{L}, \tag{3.9}$$
$$\tau_k^{i,l} \geq W_{j,k}^i, \quad j,k \in \mathcal{M}^i, e_{j,k}^i \in \mathcal{E}^i, i \in \mathcal{N}, l \in \mathcal{L}, \tag{3.10}$$
$$x_0^{i,0} = 1, x_{M^i}^{i,0} = 1, \tau_0^{i,0} = \delta^i, \quad i \in \mathcal{N}. \tag{3.11}$$

Constraint (3.2) holds that a task can be computed on either a mobile device or the server. If task j of application i is decided to be computed on mobile device i, $x_j^{i,0} = 1$. Otherwise, if task j of application i is decided to be offloaded to the server, $x_j^{i,0} = 0$ and $\sum_{l \in \mathcal{L} \setminus \{0\}} x_j^{i,l} = 1$. Constraint (3.3) shows the relationship between the beginning and the ending times of the processing of a segment of task j of application on core l, where $x_j^{i,l} C_j^i / f_l$ is the processing time on core l. Constraint (3.4) states that task j of application i can be sequentially processed on various cores if necessary, i.e., different segments of the task must be processed in different time slices. Constraint (3.5) holds that a core on the server can process only one task at most at a time. Constraint (3.6) shows that $y_{j,k}^i$ is determined by $x_j^{i,0}$ and $x_k^{i,0}$. Constraint (3.7) shows the relationship between the beginning and the ending times of the data transmission from task j to task k of application i where $d_{j,k}^i / s$ is the time consumption for data transmission from task j to task k of application i via the shared channel. Constraint (3.8) guarantees that no transmission collision occurs over the shared channel. Constraint (3.9) indicates that the data from task j to its subsequent task k should be transmitted only after the computation of task j is complete. Similarly, Constraint (3.10) indicates that the processing of task k should be after than all input data from its preceding tasks $j (e_{j,k}^i \in \mathcal{E}^i)$ have been received. Constraint (3.11) shows that the first and the last tasks of an application are un-offloadedable and the execution beginning time of application i is δ^i.

Since problem (1) is a mixed-integer nonlinear programming (MINLP) prob-

lem, we rewrite problem (3.1) as a MILP problem as follows.

$$\min \qquad T = \frac{1}{N}\sum_{i\in\mathcal{N}}\left(T^{i,0}_{M^i} - \tau^{i,0}_0\right), \tag{3.12}$$

subject to

(3.2),(3.3),(3.7),(3.9)–(3.11),

$$\alpha^{i,l,l'}_j \geq (\tau^{i,l'}_j - \tau^{i,l}_j)/I, \quad j \in \mathcal{M}^i, i \in \mathcal{N}, l, l' \in \mathcal{L}, l \neq l', \tag{3.13}$$

$$1 - \alpha^{i,l,l'}_j \geq (\tau^{i,l}_j - \tau^{i,l'}_j)/I, \quad j \in \mathcal{M}^i, i \in \mathcal{N}, l, l' \in \mathcal{L}, l \neq l', \tag{3.14}$$

$$\alpha^{i,l',l}_j + \alpha^{i,l,l'}_j = 1, \quad j \in \mathcal{M}^i, i \in \mathcal{N}, l, l' \in \mathcal{L}, l \neq l', \tag{3.15}$$

$$z^{i,l}_j \geq x^{i,l}_j, \quad j \in \mathcal{M}^i, i \in \mathcal{N}, l \in \mathcal{L}, \tag{3.16}$$

$$\tau^{i,l'}_j \geq T^{i,l}_j - I(3 - \alpha^{i,l,l'}_j - z^{i,l'}_j - z^{i,l}_j),$$
$$j \in \mathcal{M}^i, i \in \mathcal{N}, l, l' \in \mathcal{L}, l \neq l', \tag{3.17}$$

$$\beta^{i,i',l}_{j,j'} \geq (\tau^{i',l}_{j'} - \tau^{i,l}_j)/I,$$
$$j \in \mathcal{M}^i, j' \in \mathcal{M}^{i'}, i, i' \in \mathcal{N}, (i,j) \neq (i',j'), l \in \mathcal{L} \setminus \{0\}, \tag{3.18}$$

$$1 - \beta^{i,i',l}_{j,j'} \geq (\tau^{i,l}_j - \tau^{i',l}_{j'})/I,$$
$$j \in \mathcal{M}^i, j' \in \mathcal{M}^{i'}, i, i' \in \mathcal{N}, (i,j) \neq (i',j'), l \in \mathcal{L} \setminus \{0\}, \tag{3.19}$$

$$1 = \beta^{i,i',l}_{j,j'} + \beta^{i',i,l}_{j',j},$$
$$j \in \mathcal{M}^i, j' \in \mathcal{M}^{i'}, i, i' \in \mathcal{N}, (i,j) \neq (i',j'), l \in \mathcal{L} \setminus \{0\}, \tag{3.20}$$

$$\tau^{i',l}_{j'} \geq T^{i,l}_j - I(3 - \beta^{i,i',l}_{j,j'} - z^{i',l}_{j'} - z^{i,l}_j),$$
$$j \in \mathcal{M}^i, j' \in \mathcal{M}^{i'}, i, i' \in \mathcal{N}, (i,j) \neq (i',j'), l \in \mathcal{L} \setminus \{0\}, \tag{3.21}$$

$$y^i_{j,k} \geq x^{i,0}_j - x^{i,0}_k, \quad j,k \in \mathcal{M}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N}, \tag{3.22}$$

$$y^i_{j,k} \geq x^{i,0}_k - x^{i,0}_j, \quad j,k \in \mathcal{M}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N}, \tag{3.23}$$

$$y^i_{j,k} \leq x^{i,0}_j + x^{i,0}_k, \quad j,k \in \mathcal{M}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N}, \tag{3.24}$$

$$y^i_{j,k} \leq 2 - x^{i,0}_j - x^{i,0}_k, \quad j,k \in \mathcal{M}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N}, \tag{3.25}$$

$$\gamma^{i,i'}_{j,k,j',k'} \geq (\omega^{i'}_{j',k'} - \omega^i_{j,k})/I, \quad j,k \in \mathcal{M}^i, j',k' \in \mathcal{M}^{i'},$$
$$e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i, i' \in \mathcal{N}, (i,j,k) \neq (i',j',k'), \tag{3.26}$$

$$1 - \gamma^{i,i'}_{j,k,j',k'} \geq (\omega^i_{j,k} - \omega^{i'}_{j',k'})/I, \quad j,k \in \mathcal{M}^i, j',k' \in \mathcal{M}^{i'},$$
$$e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i, i' \in \mathcal{N}, (i,j,k) \neq (i',j',k'), \tag{3.27}$$

$$1 = \gamma^{i,i'}_{j,k,j',k'} + \gamma^{i',i}_{j',k',j,k}, \quad j,k \in \mathcal{M}^i, j',k' \in \mathcal{M}^{i'},$$
$$e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}, (i,j,k) \neq (i',j',k'), \tag{3.28}$$
$$\omega^{i'}_{j',k'} \geq W^i_{j,k} - I(3 - \gamma^{i,i'}_{j,k,j',k'} - y^{i'}_{j',k'} - y^i_{j,k}), \quad j,k \in \mathcal{M}^i, j',k' \in \mathcal{M}^{i'},$$
$$e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}, (i,j,k) \neq (i',j',k'). \tag{3.29}$$

Proposition 1 *Problem (3.1) and problem (3.12) are equivalent.*

Proof *Note that constraints (3.4), (3.5), (3.6) and (3.8) are nonlinear; we rewrite these constraints as linear constraints to formulate problem (3.1) as a linear optimization problem as (3.12).*

For this purpose, we introduce three new binary integer variables, $z^{i,l}_j$, $\alpha^{i,l,l'}_j$ and $\beta^{i,i',l}_{j,j'}$ to help us rewrite constraints (3.4) and (3.5). $z^{i,l}_j$ represents whether a segment of task j of application i is processed by core l: $z^{i,l}_j$ is 1 if a segment of task j is processed by core l, and is 0 otherwise. $\alpha^{i,l,l'}_j$ represents the processing order of two segments of task j of application i: $\alpha^{i,l,l'}_j$ is 1 if $\tau^{i,l}_j \leq \tau^{i,l'}_j$ and is 0 otherwise. $\beta^{i,i',l}_{j,j'}$ represents the processing order of the segments of two different tasks, task j of application i and task j′ of application i′: $\beta^{i,i',l}_{j,j'}$ is 1 if $\tau^{i,l}_j \leq \tau^{i',l}_{j'}$ and 0 otherwise. Additionally, we also introduce a positive constant I that is greater than any other variable used in this study.

Then, we can rewrite constraint (3.4) as constraints (3.13)–(3.17). Constraints (3.13) and (3.14) indicate that $\alpha^{i,l,l'}_j$ must be 1 if $\tau^{i,l}_j \leq \tau^{i,l'}_j$ and 0 if $\tau^{i,l}_j \geq \tau^{i,l'}_j$. Constraint (3.15) holds that two computations should have a sequential order, i.e., $\alpha^{i,l,l'}_j \neq \alpha^{i,l',l}_j$. Constraint (3.16) indicates that $z^{i,l}_j = 1$ if a segment of task j of application i is processed by core l, i.e., $x^{i,l}_j > 0$. Constraint (3.17) indicates that if two segments of task j of application i are processed at cores l and l′, i.e., $z^{i,l}_j = z^{i,l'}_j = 1$, and the computation of the segment of this task at core l is earlier than the processing at core l′, i.e., $\alpha^{i,l,l'}_j = 1$, then the beginning time of processing at core l′ must be later than the ending time of processing at core l, i.e., $\tau^{i,l'}_j \geq T^{i,l}_j$. Constraints (3.13)–(3.15) and (3.17) guarantee that when problem (3.12) is minimized, $z^{i,l}_j = 0$ if $x^{i,l}_j = 0$.

We can rewrite constraint (3.5) similarly to constraint (3.4) as constraints (3.18)–(3.21). Note that constraints (3.18)–(3.21) do not hold for a mobile device since it has enough CPU cores.

Constraint (3.6) can be rewritten as constraints (3.22)–(3.25). These constraints ensure that $y^i_{j,k}$ is 0 if $x^{i,0}_j = x^{i,0}_k$ and is 1 otherwise.

For rewriting constraint (3.8), we need to introduce another new binary integer variable $\gamma^{i,i'}_{j,k,j',k'}$, which indicates whether the data transmission from tasks j to k of application i is faster than another data transmission from task j' to k' of application i'. $\gamma^{i,i'}_{j,k,j',k'}$ is 1 if $\omega^{i}_{j,k} \leq \omega^{i'}_{j',k'}$ and is 0 otherwise. Following the idea of rewriting constraint (3.5), we can linearly rewrite constraint (3.8) as constraints (3.26)–(3.29). ■

Problem (3.12) can be solved using an optimization solver such as Gurobi Optimizer. However, on one hand, it is extremely hard to find the optimal solution in a short time. On the other hand, in reality, the exact execution beginning time and information of an application are difficult to obtain in advance, which makes the problem even more challenging to solve. In next section, we will study how to offload applications in an online scenario.

3.3 Online Offloading Problem

Generally, an application is executed repeatedly but without any exact information about when it is executed and how much data it has to process. In this section, we consider an online scenario in which each application can be executed at any time at a mobile device. After the start of execution of an application, the mobile device firstly sends the information about the application to the edge server to obtain the offloading strategy. The server then determines the offloading strategy for the application depending on its current workload state and sends the strategy back to the mobile device. At last, the mobile device can execute the application following the determined offloading strategy. The notations used in this section is shown in Table 3.4.

Table 3.4. Notations used in online offloading problem.

Symbol	Meaning
$\mathcal{N}'$	set of applications including those currently under execution and the current newly executed application
N'	current newly executed application
$\mathcal{S}^i$	offloading strategy for application i
η	degree of system congestion
H	congestion threshold of system
$\mathcal{C}$	occupation of cores at the server

In the online scenario, when an application is newly executed, we attempt to minimize the response time for the application while simultaneously avoiding the

disturbance of the tasks under processing. Therefore, we record the offloading strategy of the applications under execution as equations (3.30) and (3.31). The newly executed application is denoted as N', the under execution applications are denoted by $\{1, 2, ..., N'-1\}$ and the set of these applications is denoted as $\mathcal{N}' = \{1, 2, ..., N'\}$. In $\mathcal{N}'$, the applications are ordered by their execution beginning time.

$$x_j'^{i,l} = x_j^{i,l}, \tau_j'^{i,l} = \tau_j^{i,l}, T_j'^{i,l} = T_j^{i,l}, \quad j \in \mathcal{M}^i, i < N', l \in \mathcal{L}, \tag{3.30}$$

$$y_{j,k}'^{i} = y_{j,k}^{i}, \omega_{j,k}'^{i} = \omega_{j,k}^{i}, W_{j,k}'^{i} = W_{j,k}^{i}, \quad e_{j,k}^{i} \in \mathcal{E}^i, j, k \in \mathcal{M}^i, i < N'. \tag{3.31}$$

The offloading strategies of the applications under execution are treated as constraints for the online offloading problem to minimize the response time of application N'. Then, the online offloading problem can be formulated as problem (3.32) for each newly executed application N' by extending problem (3.12) as follows.

$$\min \quad \frac{1}{N'} \sum_{i \in \mathcal{N}'} (T_{M^i}^{i,0} - \tau_0^{i,0}), \tag{3.32}$$

subject to

$$\text{(3.2),(3.3),(3.7),(3.9)–(3.11),(3.13)–(3.29),}$$

$$x_j^{i,l} = x_j'^{i,l}, \tau_j^{i,l} = \tau_j'^{i,l}, T_j^{i,l} = T_j'^{i,l},$$
$$j \in \mathcal{M}^i, i < N', l \in \mathcal{L}, \tau_j^{i,l} < \delta^{N'}, \tag{3.33}$$

$$y_{j,k}^{i} = y_{j,k}'^{i}, \omega_{j,k}^{i} = \omega_{j,k}'^{i}, W_{j,k}^{i} = W_{j,k}'^{i},$$
$$e_{j,k}^{i} \in \mathcal{E}^i, j, k \in \mathcal{M}^i, i < N', \omega_{j,k}^{i} < \delta^{N'}. \tag{3.34}$$

Correspondingly, $\mathcal{N}'$ is used to replace $\mathcal{N}$ in constraints (3.2),(3.3),(3.7), (3.9)–(3.11) and (3.13)–(3.29).

Now, the online offloading problem can be solved by solving problem (3.32). Unfortunately, it is still extremely hard to solve problem (3.32) in a short time. Hence, we propose an offloading strategy determination approach that solves the online offloading problem more efficiently. The overview of the determination approach is shown in Figure 3.5. We firstly find an initial offloading strategy without the considering of the limitation of the CPU cores on the server. Then, we adjust the initial offloading strategy while considering the limitation of the CPU cores to obtain the an offloading strategy. In order to obtain an initial offloading strategy, we propose two determination approach: one is based on the optimization technique (Opt. approach), and the other one is a heuristic approach.

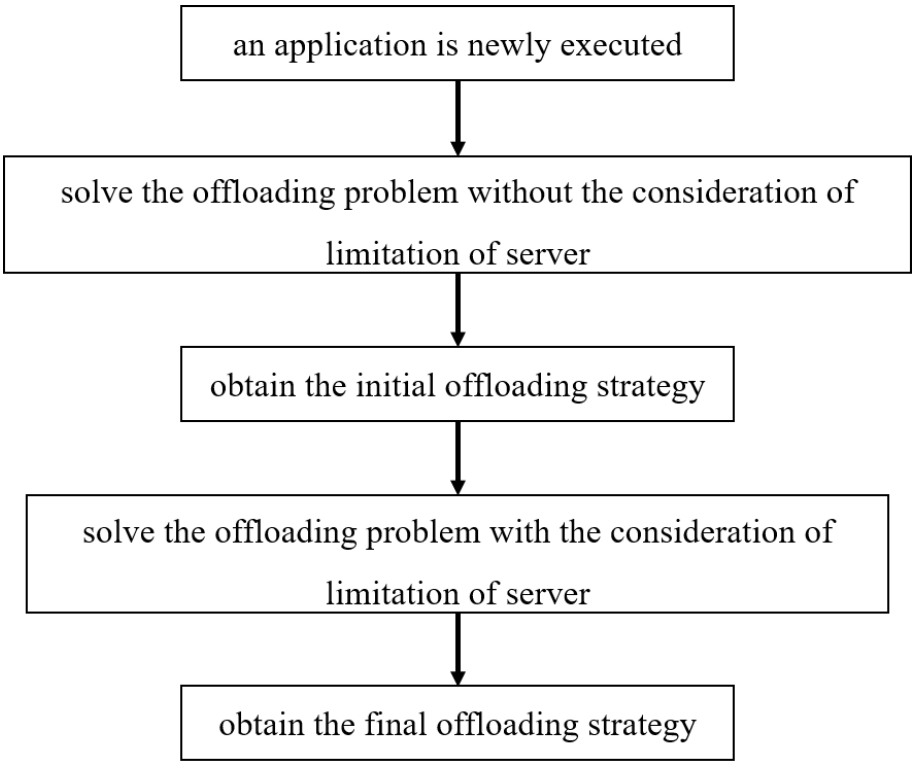


Figure 3.5. An overview of the determination approach for online offloading problem.

3.3.1 Optimization Technique based Approach to Obtain an Initial Offloading Strategy

Since the limitation of CPU cores on the server is not considered in initial offloading strategy obtainment, constraints (3.2), (3.13)–(3.21) and (3.33) can be removed. Additionally, all of offloading decisions of tasks $x_j^{i,l}$ are replaced by x_j^i, where $x_j^i = x_j^{i,0}$, and $\tau_j^{i,l}$ and $T_j^{i,l}$ can be simplified as τ_j^i and T_j^i, respectively. Furthermore, constraint (3.3) is replaced by $T_j^i - \tau_j^i = x_j^i C_j^i / f_0 + (1 - x_j^i) C_j^i / f_1,\ j \in \mathcal{M}^{N'}$, where f_1 denotes the processing rate of the server. To accelerate the convergence when solving problem, we focus only on the minimization of the response time of application N' and avoid the disturbance of the applications under execution at that time. Then, the problem is transformed into problem (3.35).

$$\min \qquad T = T_{M^i}^{N'} - \tau_0^{N'}, \tag{3.35}$$

subject to

$$T_j^{N'} - \tau_j^{N'} = x_j^{N'} C_j^{N'} / f_0 + (1 - x_j^{N'}) C_j^{N'} / f_1, \quad j \in \mathcal{M}^{N'}, \tag{3.36}$$

$$y_{j,k}^{N'} = \left| x_j^{N'} - x_k^{N'} \right|, \quad j,k \in \mathcal{M}^{N'}, e_{j,k}^{N'} \in \mathcal{E}^{N'}, \tag{3.37}$$

$$W_{j,k}^{N'} - \omega_{j,k}^{N'} = y_{j,k}^{N'} d_{j,k}^{N'}/s, \quad j,k \in \mathcal{M}^{N'}, e_{j,k}^{i} \in \mathcal{E}^{N'}, \tag{3.38}$$

$$\begin{aligned} &\left| W_{j,k}^{i} - W_{j',k'}^{i'} \right| + \left| \omega_{j,k}^{i} - \omega_{j',k'}^{i'} \right| \geq y_{j,k}^{i} d_{j,k}^{i}/s + y_{j',k'}^{i'} d_{j',k'}^{i'}/s, \\ &\qquad j,k \in \mathcal{M}^{i}, j',k' \in \mathcal{M}^{i'}, e_{j,k}^{i} \in \mathcal{E}^{i}, e_{j',k'}^{i'} \in \mathcal{E}^{i'}, \\ &\qquad N' \in \{i,i'\} \in \mathcal{N}', (i,j,k) \neq (i',j',k'), \end{aligned} \tag{3.39}$$

$$\omega_{j,k}^{N'} \geq T_j^{N'}, \quad j,k \in \mathcal{M}^{N'}, e_{j,k}^{N'} \in \mathcal{E}^{N'}, \tag{3.40}$$

$$\tau_k^{N'} \geq W_{j,k}^{N'}, \quad j,k \in \mathcal{M}^{N'}, e_{j,k}^{N'} \in \mathcal{E}^{N'}, \tag{3.41}$$

$$y_{j,k}^{i} = y_{j,k}^{\prime i}, \omega_{j,k}^{i} = \omega_{j,k}^{\prime i}, W_{j,k}^{i} = W_{j,k}^{\prime i}, \quad e_{j,k}^{i} \in \mathcal{E}^{i}, j,k \in \mathcal{M}^{i}, i < N', \tag{3.42}$$

$$x_0^{N'} = x_{M^{N'}}^{N'} = 1, \tau_0^{N'} = \delta^{N'}. \tag{3.43}$$

Constraint (3.39) can be linear rewritten similarly as Proposition 1 and problem (3.35) can be solved using an optimization solver. However, the solving time of problem (3.35) may be long when the system is congested. Thus, we also propose a heuristic approach to obtain an initial offloading strategy.

3.3.2 Heuristic Approach to Obtain an Initial Offloading Strategy

Our proposed heuristic approach consists of three sub-algorithms: a single-user chain-application (SUCA) offloading algorithm that determines the optimal offloading tasks for an application with a simple task chain, a single-user general-application (SUGA) offloading algorithm that considers only one application in the system and makes the optimal offloading decision for each task on the task flow graph shown in Figure 3.3, and a multi-user general-application (MUGA) offloading algorithm that considers multiple applications and an application may compete with another application for data transmission via the shared communication channel.

In SUCA offloading problem, we consider a simple scenario here where there is only one application in the system and the application consists of a simple task chain as shown in Figure 3.6. For the sake of simplicity, we omit i from the superscripts of the variables. That is, x_j^i, C_j^i,$d_{j,k}^i$, $e_{j,k}^i$ and $\mathcal{M}^i$ are simply denoted by x_j, $d_{j,k}$, $e_{j,k}$, $\mathcal{M}$. The SUCA offloading problem satisfies the constraints (3.3), (3.6), (3.7),(3.9)–(3.11).

We see from [55,56] that a chain application can be offloaded at most only once. Furthermore, we can show that the beginning and the ending offloaded tasks of an chain application satisfy the following theorem.

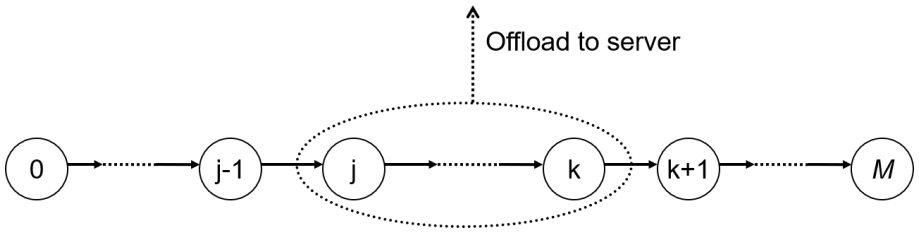


Figure 3.6. A chain application.

Theorem 1 *For a mobile application with a simple task chain from 0 to M, suppose that there exists an optimal offloading decision, with the beginning and ending tasks j^* and k^*, respectively, for offloading and that the offloading decision yields the shortest response time. Then, the optimal beginning task j^* remains unchanged no matter the ending task is k^* or $M-1$. Similarly, the optimal ending task k^* remains unchanged no matter the beginning task is j^* or 1.*

Proof *We firstly prove that by offloading tasks beginning from task j^* to $M-1$ yields a shorter response time than that beginning from any other task j. If $j \leq k^*$, we have the following inequality because tasks j^* and k^* are the optimal beginning and ending tasks.*

$$\sum_{l=0}^{j-1} C_l/f_0 + \sum_{l=j}^{k^*} C_j/f_1 + \sum_{l=k^*+1}^{M} C_l/f_0 + \frac{d_{j-1,j}}{s} + \frac{d_{k^*,k^*+1}}{s} > \sum_{l=0}^{j^*-1} C_l/f_0 + \sum_{l=j^*}^{k^*} C_l/f_1 + \sum_{l=k^*+1}^{M} C_l/f_0 + \frac{d_{j^*-1,j^*}}{s} + \frac{d_{k^*,k^*+1}}{s}. \quad (3.44)$$

That is,

$$\sum_{l=0}^{j-1} C_l/f_0 + \sum_{l=j}^{k^*} C_l/f_1 + \frac{d_{j-1,j}}{s} > \sum_{l=0}^{j^*-1} C_l/f_0 + \sum_{l=j^*}^{k^*} C_l/f_1 + \frac{d_{j^*-1,j^*}}{s}. \quad (3.45)$$

Then, by adding $\sum_{l=k^*+1}^{M-1} C_l/f_1 + C_M/f_0 + \frac{d_{M-1,M}}{s}$ to the two sides of inequality (3.45), we have

$$\sum_{l=0}^{j-1} C_l/f_0 + \sum_{l=j}^{M-1} C_l/f_1 + C_M/f_0 + \frac{d_{j-1,j}}{s} + \frac{d_{M-1,M}}{s} > \sum_{l=0}^{j^*-1} C_l/f_0 + \sum_{l=j^*}^{M-1} C_l/f_1 + C_M/f_0 + \frac{d_{j^*-1,j^*}}{s} + \frac{d_{M-1,M}}{s}. \quad (3.46)$$

From (3.46), we see that the application response time by offloading tasks from task j^* to $M-1$ is shorter than that by offloading tasks from any task j ($j \leq k^*$) to $M-1$.

On the other hand, we see from the assumption of Theorem 1 that offloading tasks from j^* to k^* results in a shorter response time than the case without offloading those tasks. That is, we have the following inequality.

$$\sum_{l=j^*}^{k^*} C_l/f_0 > \sum_{l=j^*}^{k^*} C_l/f_1 + \frac{d_{j^*-1,j^*}}{s} + \frac{d_{k^*,k^*+1}}{s}.$$

Therefore, if $j > k^*$, we have

$$\begin{aligned}
&\sum_{l=0}^{j-1} C_l/f_0 + \sum_{l=j}^{M-1} C_l/f_1 + C_M/f_0 + \frac{d_{j-1,j}}{s} + \frac{d_{M-1,M}}{s} \\
&\quad > \sum_{l=0}^{j^*-1} C_l/f_0 + \sum_{l=j^*}^{k^*} C_l/f_1 + \frac{d_{j^*-1,j^*}}{s} + \frac{d_{k^*,k^*+1}}{s} \\
&\qquad + \sum_{l=k^*+1}^{j-1} C_l/f_0 + \sum_{l=j}^{M-1} C_l/f_1 + C_M/f_0 + \frac{d_{j-1,j}}{s} + \frac{d_{M-1,M}}{s} \\
&\quad > \sum_{l=0}^{j^*-1} m_l + \sum_{l=j^*}^{M-1} c_l + m_M + \frac{d_{j^*-1,j^*}}{s} + \frac{d_{M-1,M}}{s}. \qquad (3.47)
\end{aligned}$$

Again, we see from (3.47) that the application response time by offloading tasks from task j^* to $M-1$ is shorter than that by offloading tasks from any task j ($j > k^*$) to $M-1$. Therefore, we can conclude that the optimal beginning task j^* remains unchanged even if the ending task k^* is moved to $M-1$.

Similarly, we can prove that the application response time by offloading tasks from task 1 to k^* is shorter than that by offloading tasks from task 1 to any other task k. Therefore, we can conclude that the optimal ending task k^* remains unchanged even if the beginning task j^* is moved to task 1. ■

According to Theorem 1, we can find the optimal beginning and ending offloaded tasks respectively by scanning the task chain at most twice. The following single-user chain-application (SUCA) algorithm leads to the optimal beginning and ending offloaded tasks for an application with a task chain from 0 to M. In some cases, because there might be some tasks must be offloaded, we add two variables J_0 and K_0 that the tasks from J_0 to K_0 must be offloaded. The input parameters of SUCA include C_j ($j \in \mathcal{M}$), $d_{j,j+1}$ ($j \in \mathcal{M} \setminus \{M\}$), f_0, f_1, s, J_0, and K_0. Here, the beginning and the ending offloaded tasks are chosen from the ranges of $[1, J_0]$

and $[K_0, M-1]$, respectively. If there is no tasks must be offloaded, $J_0 = M-1$ and $K_0 = 1$. The computation complexity of the SUCA algorithm is bounded by $O\Big(M\Big)$, where M is the number of tasks in the application.

Algorithm 1 Single-user chain-application (SUCA) offloading algorithm.

Input: : C_j $(j \in \mathcal{M}), \mathcal{E}, d_{j,j+1}$ $(j \in \mathcal{M} \setminus \{M\}), f_0, f_1, s$

1: calculate the response time of the application, $T_{\max}$, without offloading any task, i.e., $T_{\max} = \sum_{j=1}^{M-1} m_j$
2: calculate the response time with offloading tasks from j $(j \in [1, M-1])$ to $M-1$ and determine task j^* that yields the shortest response time
3: calculate the response time with offloading tasks from 1 to k $(k \in [1, M-1])$ and determine task k^* that yields the shortest response time
4: calculate the response time T_{j^*,k^*} with the beginning and ending offloaded tasks j^* to k^*
5: **if** $T_{j^*,k^*} \geq T_{\max}$ **then**
6: **return** null
7: **else**
8: **return** j^*, k^*
9: **end if**

After the SUCA offloading problem, we consider the case of only one application whose tasks can be processed parallelly and/or sequentially as shown in Figure 3.3 but the collision of data transmission is not taken into account (SUGA problem). We also omit i here from the symbols used for the variables. Note that the constraints (3.3), (3.6), (3.7) and (3.9)–(3.11) are also satisfied here.

The offloading strategy for SUGA are based on the following two considerations.

1. In order to minimize the response time of an application we need to minimize the path length from task 0 to M on the task flow graph. The length of a path means the sum of the task execution times and the transmission delays along the path from task 0 to M on the task flow graph.

2. Since a path from task 0 to M can be considered as a task chain, the tasks on any path can be offloaded only once and the offloaded tasks should be consecutive. Therefore, the tasks on any bypass of a path that goes out from an offloaded task and back to another offloaded task on a path should also be offloaded. To understand this idea better, let us see an example shown in Figure 3.7 where the number shown above each node denotes the task

execution time. We see that the longest path passes through tasks 0, 1, 4, 8, 10. To minimize the response time of this application, we need to offload some tasks on the longest path to the server. Suppose that we determine to offload tasks 1, 4, and 8 to the server and then we need to examine whether there is any bypass of the longest path. We can find a bypass along the longest path that goes out from task 1 to 8 through task 5 and therefore task 5 should also be offloaded.

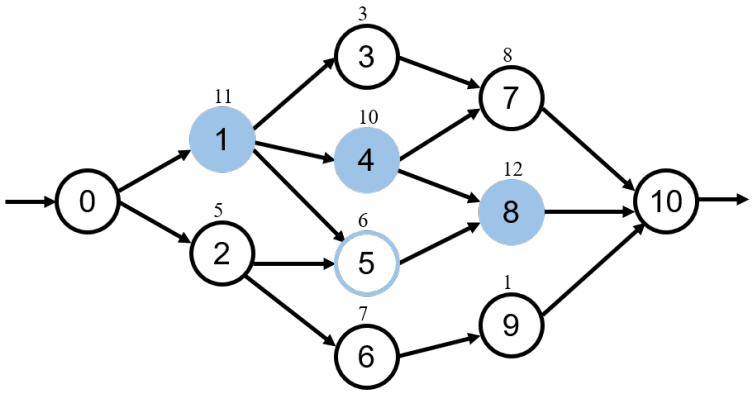


Figure 3.7. An example of a task flow graph.

The single-user general-application (SUGA) algorithm is described in Algorithm 2 in details. In each iteration, we decide at least one task that should be offloaded, and therefore the computation complexity of SUGA is bounded by $O\left(M^3\right)$, where M is the number of tasks in the application.

Then, we consider the case of that there are multiple applications in the system, and only one data transmissions can happen at any time instant (MUGA problem). The MUGA offloading problem satisfies the constraints (3.3), (3.6)–(3.11). When the application N' is newly executed, the MUGA algorithm determines the offloading strategy for application N' and, if necessary, updates the offloading strategies for other applications i $(i < N')$ that have been executed earlier than application N'.

The MUGA algorithm consists of the following three phases.

1. Initial offloading decision of tasks in application N'.

   We determine the initial offloading decisions of tasks in application N' using **Algorithm 2** and calculate the offloading strategy of application N' as $\mathcal{S}^{N'}$ without considering the collision over the shared channel.

2. Elimination of interruption to existing application.

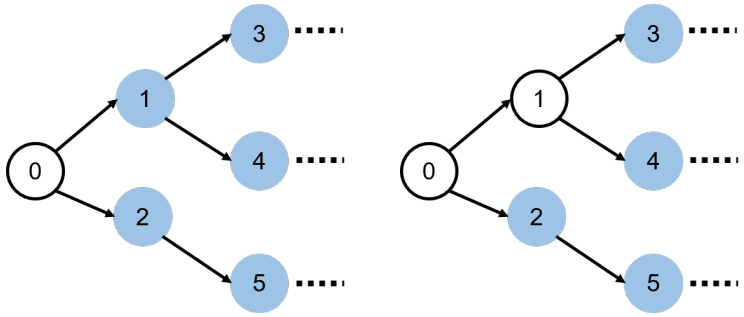
Algorithm 2 Single-user general-application (SUGA) offloading algorithm.

Input: : C_j $(j \in \mathcal{M}), \mathcal{E}, d_{j,k}$ $(j,k \in \mathcal{M}, e_{j,k} \in \mathcal{E}), f_0, f_1, s$

let $L = \{\mathcal{M}\}, C = \emptyset$ and $x_j = 1$ $(j \in \mathcal{M})$
calculate the response time of the application as $T_{\max}$
repeat
  find the current longest path p from task 0 to M
  if $x_j = 0$ such that task j is on path p **then**
    set the first and the last offloaded tasks along p to be J_0 and K_0, respectively
  else
    set $J_0 = 0$ and $K_0 =$ the previous task to M along p
  end if
  determine the beginning and ending tasks for offloading on path p by **Algorithm 1**
  set $x_j = 0$ for each task on the bypass of path p from j^* to k^* and let $C = C \cup j, L = L \setminus j$
  calculate the current response time as T_p
  if $T_p > T_{max}$ **then**
    Let $x_j = 1$ $(j \in \mathcal{M})$ and **break**
  end if
  for each bypass p_{ij} from an offloaded task i to j $(i, j \in C)$ **do**
    if task k on path p_{ij} and $k \in L$ **then**
      Let $x_k = 0$ and $C = C \cup k, L = L \setminus k$
    end if
  end for
until no new offloading task can be found
return $x_0, x_1, \cdots, x_M$

We consider the collision over the shared channel that a data transmission of application N' may collide with another transmission of existing applications i $(i < N')$. Therefore, we need to determine whether to postpone the collided data transmissions of application N' or change the offloading decisions. If a data transmission of application N' from task j to k with the earliest data transmission time $\omega_{j,k}^{N'}$ collides with another data transmission of existing application i from task j' to k', we compare the response time when offloading task k, denoted as $T + W_{j',k'}^{i} - \omega_{j,k}^{N'}$ where T is the response time of application N' without considering the collision of data transmission from task j to task k, with that when not offloading task k, denoted as T'. If $T' \geq T + W_{j',k'}^{i} - \omega_{j,k}^{N'}$, the offloading decision will not be changed; other-

wise, task k will be processed locally and schedule $\mathcal{S}^i$ will be updated. See an example as shown in Figure 3.8(a) where all the tasks except task 0 of application N' are determined to be offloaded. Suppose that the data transmissions from task 0 to tasks 1 and 2 collide with other existing applications, and that the transmission from task 0 to 2 can be postponed but the transmission to task 1 cannot. Then, task 1 will be changed to process locally but the offloading decision of task 2 is not changed as Figure 3.8(b). Once we find a data transmission can be postponed, adjustment will be terminated. We describe the algorithm of offloading decision adjustment in Algorithm 3 in detail. The computation complexity is bound by $O\left(M^3\right)$ where M is the number of tasks of application N'.



(a) Initial offloading decision (b) Adjusted offloading decision

Figure 3.8. Elimination of interruption to application N'

3. Scheduling colliding transmissions.

   The data transmissions of all the applications to and back from the edge server are examined and scheduled based on a first-in-first-out (FIFO) basis. Furthermore, the data transmission of under execution applications ($\mathcal{N}' \setminus \{N'\}$) from mobile devices to the edge server should be unchanged. Additionally, if a task needs to transmit data to more than one child task via the shared channel, we need to determine to which child the data should be transmitted first. For example, if task j of application i needs to send data to two child tasks k and k', we need to determine to which child the data should be transmitted first. In our consideration, the longer one in two paths from task j to M^i via tasks k and k' respectively will be chosen to transmit data first. The algorithm of colliding transmissions scheduling is described in Algorithm 4 in detail. The computation complexity is bounded by $O\left({M_1}^2 M_2\right)$, where M_1 is the largest number of tasks of an application in $\mathcal{N}'$, and M_2 is

Algorithm 3 Offloading decision adjustment algorithm.

Input: : C_j^i $(j \in \mathcal{M}^{N'}), \mathcal{E}^{N'}, \mathcal{E}^{N'}, d_{j,k}^i$ $(j,k \in \mathcal{M}^{N'}, e_{j,k}^i \in \mathcal{E}^{N'}), f_0, f_1, s, \mathcal{S}^i$ $(i \in \mathcal{N}')$

while any $x_l^{N'} = 0$ $(l \in \mathcal{M}^{N'})$ **do**
  calculate the response time of application N' as T
  find the earliest beginning time of data transmission for application N', denoted as $\omega_{j,k}^{N'}$.
  find the last ending time for data transmission from task j' to k', $W_{j',k'}^i$, for application i $(i < N')$ such that $\tau_{j,k}^i \leq \omega_{j,k}^{N'} < W_{j',k'}^i$
  if application i exists **then**
    calculate the response time of application i, denoted by T', with $x_k^{N'} = 1$
    if $T' \geq T + W_{j',k'}^i - \omega_{j,k}^{N'}$ **then**
      break from **while** loop
    else
      $x_k^i = 1$ and update $\mathcal{S}^i$
    end if
  else
    break from **while** loop
  end if
end while
return $\mathcal{S}^{N'}$

the number of data transmissions over the channel of the applications in $\mathcal{N}'$.

In summary, MUGA algorithm is shown in Algorithm 5.

3.3.3 Core Assignment

The initial offloading strategy is obtained by means of either the Opt. approach or the MUGA algorithm without considering the limitation of CPU cores of the server, as shown in Figure 3.9. We assume that the server monitors each execution of an application and records the time for each data transmission between two tasks. The congestion degree $\eta = t'/t$ is used to select the MUGA or Opt. approach to determine the initial offloading strategy, where t denotes the estimated response time of the new application N' executed locally, and t' denotes the estimated total transmission time during the time period of the execution of application N' at the mobile device. We set the congestion threshold as H, which is a constant parameter. Therefore, the MUGA algorithm is chosen if the system is congested, i.e., $\eta > H$, and the Opt. approach is selected otherwise.

Algorithm 4 Colliding transmissions scheduling algorithm.

Input: : $\mathcal{E}^{N'}, \mathcal{S}^i \ (i \in \mathcal{N}')$

while $\omega^i_{j,k} \leq \omega^{i'}_{j',k'} < W^i_{j,k} \ (i, i' \in \mathcal{N}', \ (i,j,k) \neq (i',j',k'))$ and $\omega^{i'}_{j',k'} \neq W^{i'}_{j',k'}$ **do**
  if $x^i_k = 1$ and $i < N'$ **then**
    set $\omega^{i'}_{j',k'} = W^i_{j,k}$ and update $\mathcal{S}^{i'}$
  else if $x^{i'}_{k'} = 1$ and $i' < N'$ **then**
    set $\omega^i_{j,k} = W^{i'}_{j',k'}$ and update $\mathcal{S}^i$
  else
    if $T^i_j < T^{i'}_{j'}$ **then**
      set $\omega^{i'}_{j',k'} = W^i_{j,k}$ and update $\mathcal{S}^{i'}$
    else if $T^{i'}_{j'} < T^i_j$ **then**
      set $\omega^i_{j,k} = W^{i'}_{j',k'}$ and update $\mathcal{S}^i$
    else
      calculate the response times of applications i and i' from task j to M^i via k (denoted by T) and from task j' to $M^{i'}$ via k' (denoted by T')
      if $T < T'$ **then**
        set $\omega^{i'}_{j',k'} = W^i_{j,k}$ and update $\mathcal{S}^{i'}$
      else
        set $\omega^i_{j,k} = W^{i'}_{j',k'}$ and update $\mathcal{S}^i$
      end if
    end if
  end if
end while
return $\mathcal{S}^i \ (i \in \mathcal{N}')$

Algorithm 5 MUGA offloading algorithm.

Input: : $C^i_j \ (j \in \mathcal{M}^{N'}), d^i_{j,k} \ (j,k \in \mathcal{M}^{N'}, e^i_{j,k} \in \mathcal{E}^{N'}), f_0, f_1, s, \mathcal{S}^i \ (i \in \mathcal{N}')$

1: run **Algorithm 2** to obtain the initial offloading decision for application N' and update $\mathcal{S}^{N'}$
2: run **Algorithm 3** to adjust the offloading decision for application N' and update $\mathcal{S}^{N'}$
3: run **Algorithm 4** to schedule the data transmission for applications $\mathcal{N}'$ and update $\mathcal{S}^i \ (i \in \mathcal{N}')$
4: **return** $\mathcal{S}^i \ (i \in \mathcal{N}')$

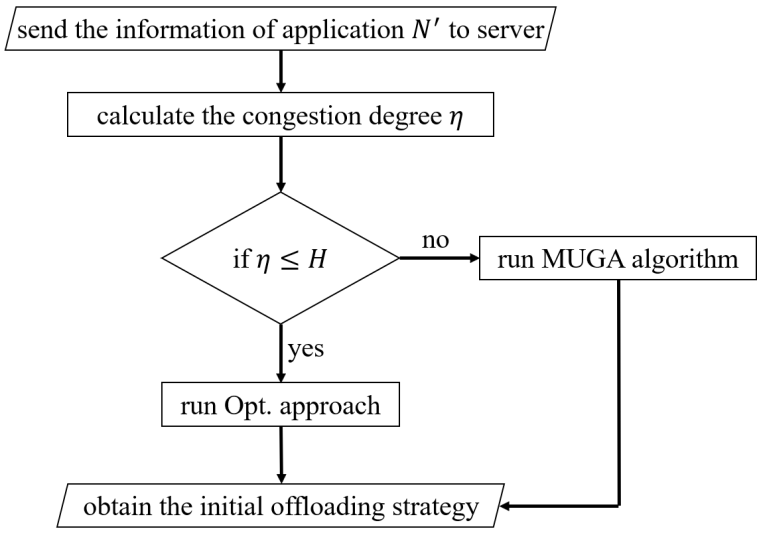
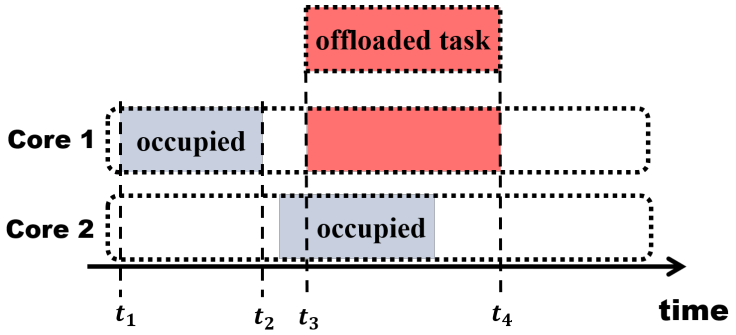


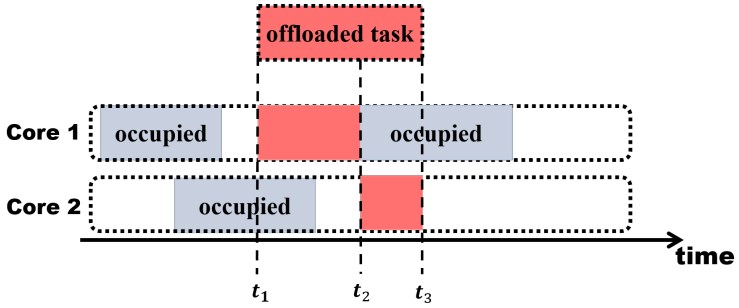
Figure 3.9. Generating an initial offloading strategy.

After obtaining the initial offloading strategy for application N', we assign the server CPU core for each of its offloaded tasks. However, it is important not to allow this assignment disturb any task processing of applications already under execution i ($i \in \mathcal{N}' \setminus \{N'\}$). That is, an offloaded task of application N' will be assigned to the unoccupied time slices on the server cores, as shown in Figure 3.10. For example, as shown in Figure 3.10(a), core 1 is occupied only from time t_1 to t_2 and task j of application N' can be assigned from time t_3 to t_4. In this case, because the whole task j is processed at core 1, the values of offloading decision of this task, i.e., $\{x_j^{N',0}, x_j^{N',1}, x_j^{N',2}\}$, are $\{0, 1, 0\}$. By contrast, if task j of application N' is offloaded and the computation is from t_1 to t_3, as shown in Figure 3.10(b), task j has to be assigned to multiple unoccupied time slices. In this case, the segment of task j from t_1 to t_2 is assigned to core 1, and the remaining task from t_2 to t_3 is assigned to core 2. Therefore, $\{x_j^{N',0}, x_j^{N',1}, x_j^{N',2}\}$ are $\{0, (t_2 - t_1)/(t_3 - t_1), (t_3 - t_2)/(t_3 - t_1)\}$.

When no free core is available at the server, task j of the newly executed application N' may be postponed, as shown in Figure3.11. There are only two cores on the server, and task j of application N' needs to be computed from t_1 to t_4 determined by initial offloading strategy. However, both of these two cores are occupied from t_2 to t_3, which means the server is overloaded. Therefore, beginning time of the residual segment of the task need to be postponed from t_2 to t_3 and the processing time becomes $t_5 - t_3$. This process leads to an additional delay in the task processing and may further postpone the processing of descendant tasks and eventually postpone the response time of the newly executed application N'. Therefore,



(a) Single assignment.



(b) Multiple successive assignment.

Figure 3.10. Core assignment for a task of a newly executed application.

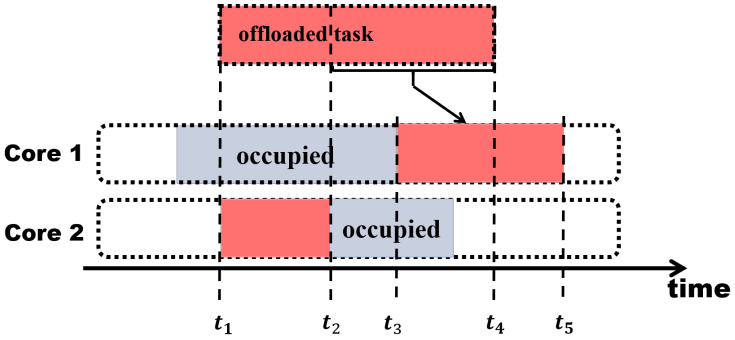


Figure 3.11. Multiple discrete assignment.

if no free cores are available for task j, we need to iteratively recalculate the beginning and ending times of the descendant tasks of task j. Moreover, since the postponement of descendant tasks of task j may affect the transmission time from the server back to the mobile device, the transmission time need to be updated and the application response time is recalculated as T'. For example in Figure 3.11, the offloading decision of task j is $\{0, (t_5 - t_3)/(t_4 - t_1), (t_2 - t_1)/(t_4 - t_1)\}$, the

beginning time of the processing of task j, i.e., $\tau_j^{N'}$, is t_1, and the ending time, i.e., $T_j^{N'}$, is t_5. The beginning and ending times of the processing of descendant tasks can be calculated by Equations (3.48) and (3.49), where j' is the descendant task of task j descendant task. Since the cores at the server are homogeneous, we use f_1 to represent the processing rate of a core at the server similar to Opt. approach and MUGA algorithm.

$$T_{j'}^{N'} = x_{j'}^{N',0} * C_{j'}^{N'} / f_0 + (1 - x_{j'}^{N',0} * C_{j'}^{N'} / f_1) + \tau_{j'}^{N'}, \tag{3.48}$$

$$\tau_{j'}^{N'} = \max\{W_{k,j'}^{N'}\},\ k \in \mathcal{N}', e_{k,j'}^{N'} \in \mathcal{E}^{N'}. \tag{3.49}$$

The data transmission time can be calculated by Equations (3.50) and (3.51), where j' is either task j or the descendant task of task j.

$$W_{j',k}^{N'} = \left| x_{j'}^{N',0} - x_k^{N',0} \right| * d_{j',k}^{N'} / s + \omega_{j',k}^{N'}, \tag{3.50}$$

$$\omega_{j',k}^{N'} = T_{j'}^{N'},\ k \in \mathcal{N}', e_{j',k}^{N'} \in \mathcal{E}^{N'}. \tag{3.51}$$

The execution schedule of task j and its descendant tasks can be iteratively calculated by Equations (3.48)–(3.51), and the response time of application N' is denoted as $T' = T_{M^{N'}}^{N'} - \tau_0^{N'}$.

Note that if the server is overloaded as shown in Figure 3.11, The benefit of offloading task j may be lower than that of processing it at the mobile device. Furthermore, if the task j is changed to be processed at local, its descendant tasks will also be processed locally. In this case, we need recalculate the response time of application N' without considering the channel bandwidth limitation. We change the offloading decision of task j as $\{1, 0, 0\}$, and calculate the beginning and ending times of the processing of its descendant tasks by Equations (3.52) and (3.53), where j' represents either task j or the descendant task of task j.

$$T_{j'}^{N'} = C_{j'}^{N'} / f_0 + \tau_{j'}^{N'}, \tag{3.52}$$

$$\tau_{j'}^{N'} = \max\{W_{k,j'}^{N'}\},\ k \in \mathcal{N}', e_{k,j'}^{N'} \in \mathcal{E}^{N'} \tag{3.53}$$

We calculate the data transmission time by Equations (3.54) and (3.55), where j' is either task j or the descendant task of task j.

$$W_{k,j'}^{N'} = (x_k^{N',0} - x_{j'}^{N',0}) * d_{k,j'}^{N'} / s + \omega_{k,j'}^{N'}, \tag{3.54}$$

$$\omega_{k,j'}^{N'} = T_k^{N'},\ k \in \mathcal{N}', e_{k,j'}^{N'} \in \mathcal{E}^{N'} \tag{3.55}$$

The execution schedule of task j and its descendant tasks in this case can be iteratively calculated by Equations (3.52)–(3.55), and the computation time of application N' in this case is denoted as $T'' = T_{M^i}^{N'} - \tau_0^{N'}$.

If $T'' < T'$, it is beneficial to process task j and its descendant tasks locally rather than offload them. We iteratively check each offloaded task and try to assign the required cores determined by the initial offloading strategy for application N'. The algorithm of core assignment is shown in Algorithm 6 in detail. Note that the core occupation information, C, is update every the execution of core assignment algorithm. $Flag$ represents whether the data transmission times in the initial offloading strategy should be updated due to the core assignment. The computation complexity is bounded by $O\left(M^3\right)$, where M is the number of tasks of application N'.

Algorithm 6 Core assignment algorithm.

Input: : $C_j^{N'}, \mathcal{E}^{N'}, d_{j,k}^{N'}\ (j,k \in \mathcal{N}^{N'}, e_{j,k}^{N'} \in \mathcal{E}^{N'}), f_l\ (l \in \mathcal{L}), s, \mathcal{C}, \mathcal{S}^{N'}$

let $CA = \emptyset$, $Flag = 0$
add the tasks of application N' that are offloaded to CA
while $CA \neq \emptyset$ **do**
  sort CA based on the beginning time of the processing of each task in CA
  assign the computation workload of the first element of CA, task j^*, to cores at the server
  for $j' \in \{$descendant tasks of $j^*\}$ and $x_{j'}^{N',0} = 0$ **do**
    recalculate $\tau_{j'}^{N'}$ and $T_{j'}^{N'}$ by Equations (3.48) and (3.49), and update the execution schedule of task j'
  end for
  while $T_{j'}^{N'} > \omega_{j',k}^{N'}$, $W_{j',k}^{N'} \neq \omega_{j',k}^{N'}\ (e_{j',k}^{N'} \in \mathcal{E}^{N'})$ **do**
    set $Flag = 1$
    recalculate the execution schedule as $\mathcal{S}'^{N'}$ and application response time T' by Equations (3.48)–(3.51)
    set $x_{j'}^{N',0} = 1$ ($j' \in \{j^*$and its descendant tasks$\}$)
    recalculate the execution schedule as $\mathcal{S}''^{N'}$ and the application response time T'' by Equations (3.52)–(3.55)
    if $T'' < T'$ **then**
      remove descendant tasks of j^* from CA and update $\mathcal{S}^{N'}$ to $\mathcal{S}''^{N'}$
    else
      update $\mathcal{S}^{N'}$ to $\mathcal{S}'^{N'}$
    end if
  end while
  $CA = CA \setminus \{j^*\}$ and update $\mathcal{C}$
end while
return $\mathcal{C}, \mathcal{S}^{N'}$ and $Flag$

If the data transmission time has been changed by the core-assignment algorithm ($Flag = 1$), the transmission order should be rescheduled. We need to recalculate the transmission times of the data transmitted from the server to the mobile devices (also called down-going data) for applications $\mathcal{N}' \setminus \{N'\}$ by Equations (3.54) and (3.55), and update $\mathcal{S}^i$ $(i \in \mathcal{N}' \setminus \{N'\})$ by Equations (3.52) and (3.53). Then, we use algorithm 4 to reschedule the data transmission time. However, it is important not to change the data transmission time of newly executed application N' from mobile devices to the edge server. Thus, we remove the judgement of $i < N'$ from steps 2 and 4 in algorithm 4.

In summary, our proposed task level offloading algorithm is shown in algorithm 7.

Algorithm 7 MUGA offloading algorithm.

Input: : C_j^i $(j \in \mathcal{M}^{N'}), \mathcal{E}^{N'}, d_{j,k}^i$ $(j,k \in \mathcal{M}^{N'}, e_{j,k}^i \in \mathcal{E}^{N'}), f_0, f_1, s, \mathcal{S}^i$ $(i \in \mathcal{N}'), H$

1: calculate the degree of system congestion η
2: **if** $\eta \leq H$ **then**
3: obtain the initial offloading strategy using Opt. approach
4: **else**
5: obtain the initial offloading strategy using MUGA algorithm (**Algorithm 5**)
6: **end if**
7: run **Algorithm 6** to assign the core for offloaded tasks and adjust the offloading strategy if necessary
8: **if** $Flag = 1$ **then**
9: recalculate the transmission times of down-going data for applications $\mathcal{N}' \setminus \{N'\}$ by Equations (3.54) and (3.55), and update $\mathcal{S}^i$ $(i \in \mathcal{N}' \setminus \{N'\})$ by Equations (3.52) and (3.53)
10: reschedule the data transmission time by running **Algorithm 4** (remove $i < N'$ from steps 2 and 4)
11: **end if**
12: **return** $\mathcal{S}^i$ $(i \in \mathcal{N}')$

3.4 Performance Evaluation

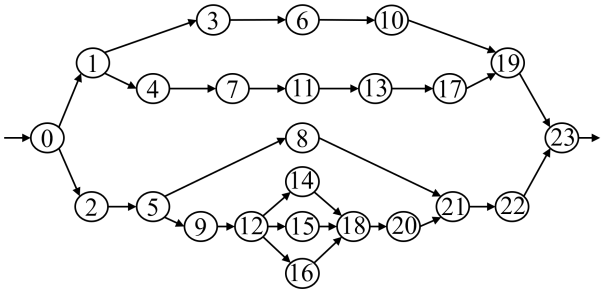
In this section, we present the performance evaluation of our proposed task level offloading algorithm, denoted by *TLO* in figures. An extreme case in which no application is offloaded, denoted by *None offloaded* in figures, is also simulated for comparison. Two previous algorithms proposed in [40] and [46], denoted by *Partial offloaded* and *Collision adjustment* in figures, are compared with our proposed

algorithm. In Partial offloaded algorithm [40], the tasks of an application are partitioned into two sets, the offloaded task set and the locally executed task set, and attempts to find the optimal partition for which the sum of the total task computing time and data transmission time is minimized. However, they do not consider the interplay between two applications so that their algorithm is hard to be used in an online scenario. To avoid causing any bias in the Partial offloaded algorithm, we assume that the exact transmission rate and the core processing rate at each time instant are known in order to make Partial offloaded algorithm adapt the online scenario. In Collision adjustment algorithm [46], an application is segmented into a set of independent tasks that can be independently offloaded to the server and formulates the problem as a MINLP problem. A suboptimal solution is found based on the optimal solution to a relaxed version of the original problem. However, an application is usually segmented into a set of dependent tasks as [40] so that their algorithm may become failure in our task level offloading problem. In order to ensure the effectiveness of Collision adjustment algorithm, each application is treated as an inseparable job in our experiments.

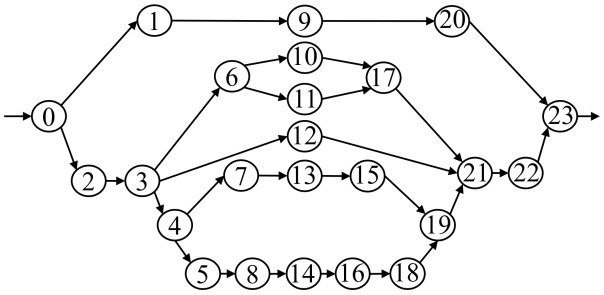
3.4.1 Parameter Settings

The setting of parameters is shown in Table 3.5. We assumed that each mobile device has the same computation power and randomly executes either a face recognition application [40] or an augmented reality application [57].The task flow graphs of these two applications are shown in Figure 3.12 and all the tasks, except tasks 0 and 23, are offloadable.

The ratio of the computing power of the server to that of a mobile device, f_l/f_0 $(l \in \mathcal{L})$, is fixed at 14, which is verified in [58]. We have tested the applications by inputting various pictures with 512×512 pixels using a computer and the average response time is approximately 115ms. Therefore, the computation time of a task on the mobile device, C_j^i/f_0 $(j \in \{1, 22\})$, is generated randomly from the range [80,240]ms, except of those of the input and tasks, C_j^i/f_0 $(j \in \{0, 23\})$, which are set to 0. Then, the data size sent between two tasks is generated randomly from the range [50,200]kB. The transmission rate, s, is set as 80Mbps. The number of mobile devices is set as 100 and each mobile device execute its application 10 times. In each mobile device, the execution of applications follows a Poisson process, i.e., the time between consecutive executions of applications on a mobile device (the inter-execution time) follows an exponential distribution. The inter-execution time of on a mobile device is set as 10s. The congestion threshold H is set as 0.4 and the number of cores of the server is set as 8. The simulation program was developed using Python 3.6, and the optimization problem was solved using the Gurobi Optimizer 8.0 on a computer running Microsoft Windows 10 with an



(a) Face recognition application.



(b) Augmented reality application.

Figure 3.12. Task flow graphs.

Intel E3 3.0 GHz CPU and 24 GB of memory.

Table 3.5. Parameter settings in experiments.

Setting	Description
100	number of mobile devices
10	application execution times on a mobile device
10s	inter-execution time of applications on a mobile device
refer to Figure 3.12	task flow of an application
$[80, 240]$ms	computation time of a task on a mobile device
$[50, 200]$kB	data size transmitted between two tasks
80Mbps	transmission rate
14	ratio of the computing power of server vs. mobile device
0.4	congestion threshold of system
8	number of cores on the server

3.4.2 Effect of Transmission Rate

Figure 3.13 shows the average application response times achieved with the different approaches for various transmission rates of the shared channel. The transmission rates are set as 0, 20, 40, 60, 80, 100, 120,140 and 160 Mbps in each experiment, respectively. Figure 3.13 shows that our proposed algorithm outperforms the other algorithms for a wide range of transmission rates. When the transmission rate is

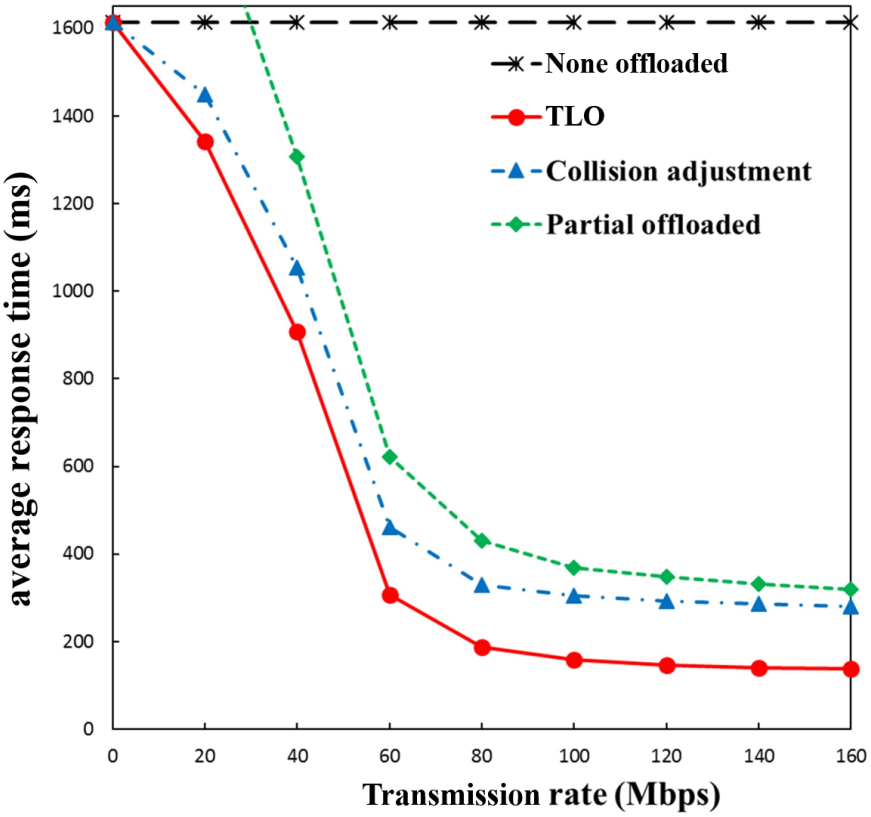


Figure 3.13. Effect of transmission rate.

low, there are few tasks which are decided to be offloaded. As the transmission rate increases, more and more tasks will be decided to be offloaded to the server. We find that when the transmission rate is low, being different with Collision adjustment algorithm and our proposed algorithm, Partial offloaded algorithm yields worse results than None offloaded. The main reason for this finding is that Partial offloaded algorithm does not consider the collisions between different data transmission over the shared channel. Therefore, the offloading strategy determined by Partial offloaded algorithm become incorrect. As the transmission rate increases,

the performance of Partial offloaded algorithm approaches that of Collision adjustment algorithm and our proposed algorithm. However, our proposed algorithm is always better than these previous algorithms because both of Partial offloaded algorithm and Collision adjustment algorithm do not consider the parallel processing of parallel tasks. Thus, utilization of the cores on the server in our offloading strategy is more efficient than that in these previous algorithms.

3.4.3 Effect of Ratio of the Computing Power of Server vs. Mobile Device

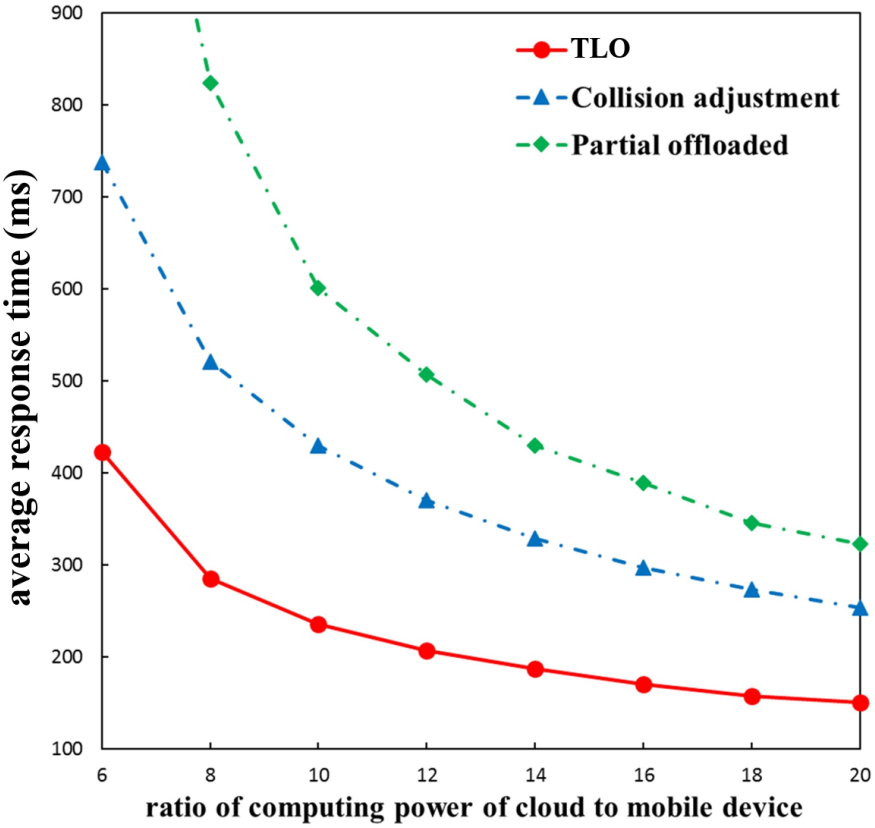


Figure 3.14. Effect of ratio of the computing power of server vs. mobile device

Figure 3.14 shows the average application response times for various values of the ratio of the computing power of the server to that of a mobile device. The values of the ratio of the computing power are set as 6, 8, 10, 12, 14, 16, 18 and 20 in each experiment, respectively. The average application response time decreases as the computing power ratio increases. We can find that the gap between our

proposed algorithm and the previous algorithms also decreases as the computing power ratio increases. It is because of that the core utilization in our offloading strategy leads to better core utilization efficiency than other algorithms. These results also show that it is important to utilize the characteristic of parallel tasks.

3.4.4 Effect of Application Inter-execution Time

Figure 3.14 shows the average application response times of the algorithms for various values of inter-execution time. The values of inter-execution time of applications on a mobile device are set as 4, 6, 8, 10, 12, 14 s in each experiment, respectively. Figure 3.15 shows that our proposed algorithm outperforms the other algorithms for a wide range of values of inter-execution time. Our proposed algorithm

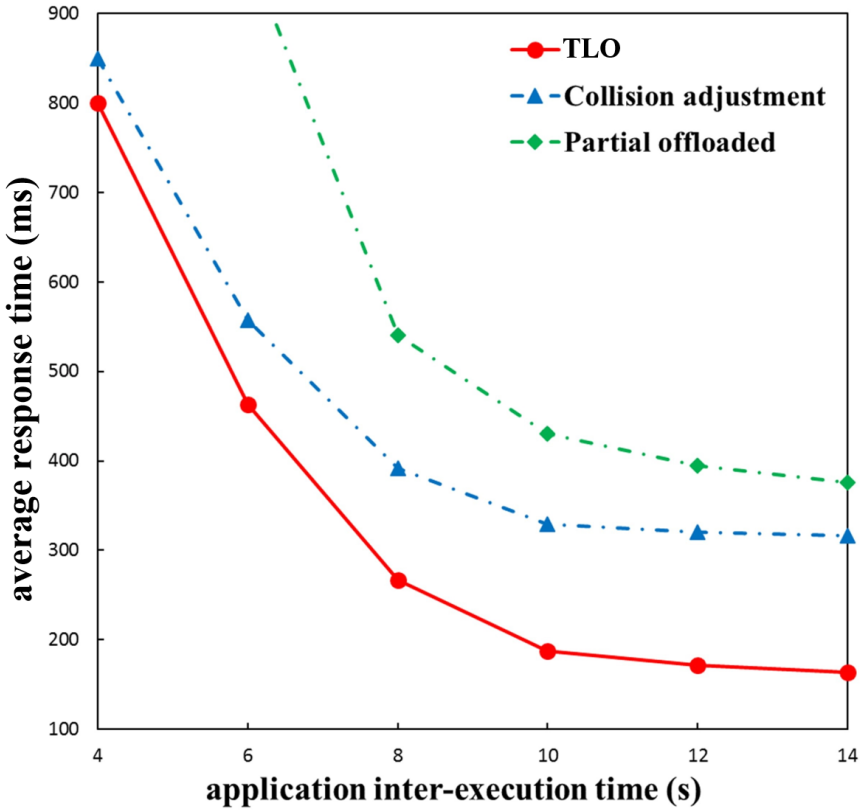


Figure 3.15. Effect of application inter-execution time.

performs closely to Collision adjustment algorithm when the inter-execution time is short. This result can be attributed to the fact that when the inter-execution time is extremely short, parallel processing is only slightly superior to serial processing. In essence, the occupation of the cores by two parallel tasks will not change

regardless of whether parallel processing or serial processing is used. However, free time slots are better utilized in parallel processing than in serial processing, and consequently, our algorithm ultimately outperforms the Collision adjustment algorithm. Since both our algorithm and Collision adjustment algorithm adjust for collisions, both are significantly superior to Partial offloaded algorithm when the inter-execution time is short. We also find that Collision adjustment algorithm performs closely to Partial offloaded algorithm when the application inter-execution time is long because when the inter-execution time is sufficiently long, there are few collisions.

3.4.5 Effect of Number of Users

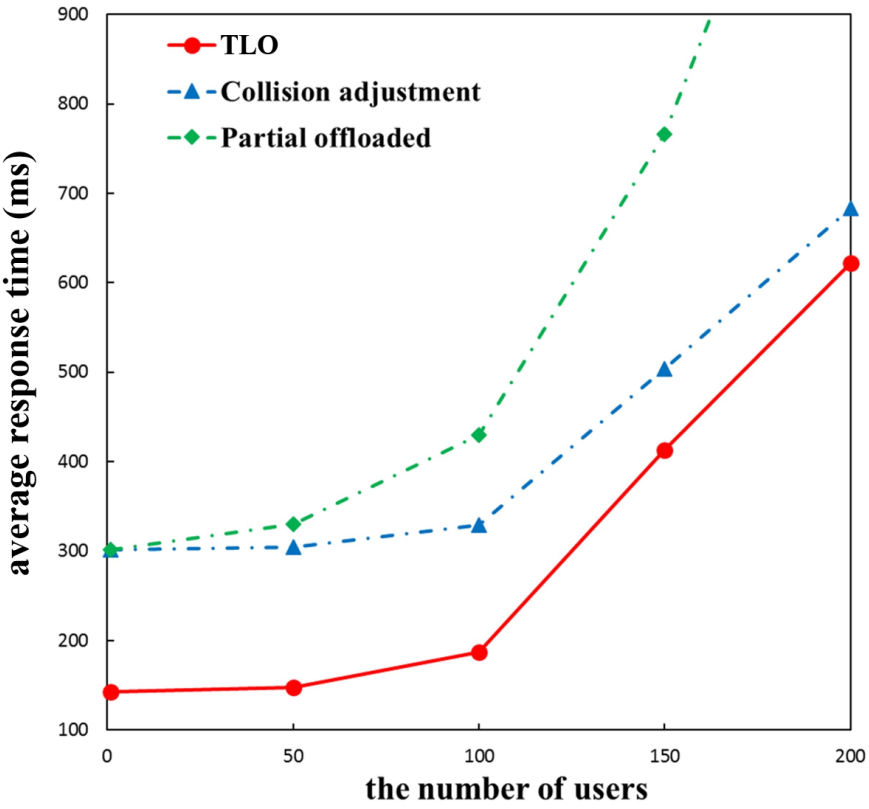


Figure 3.16. Effect of number of users.

Figure 3.16 shows the average application response times of the algorithms with various numbers of users. The numbers of users are set as 1, 50, 100, 150 and 200 in each experiment, respectively. As the number of users increases, the average application response time becomes longer since the system gradually becomes

congested. We find that our algorithm still outperforms the others even though the system becomes congested. These results prove that our algorithm shows better scalability than the other algorithms do.

3.5 Conclusion

In this chapter, we have studied the problem of offloading computationally intensive tasks of mobile applications to minimize the average application response time. We consider a number of mobile devices executing general applications connected to an edge server via a shared communication channel. We also consider the bandwidth limitation of the communication channel and the limited number of CPU cores at the server. We formulate the offline offloading problem as a MILP problem. Then, we extend the offline problem to an online offloading problem in which an application can be executed at any time. Since the online offloading problem cannot be solved in a short time, a three-phase method is proposed to determine the optimal offloading strategy for each application and the computational complexity of offloading strategy determination is substantially reduced. We firstly focus on the offloading problem while only considering the limitation of communication resource of the shared communication channel to obtain the initial offloading strategy for a newly executed application. Two effective approaches, Opt. approach and MUGA algorithm, are proposed in this phase and switched depending on the degree of system congestion: when the communication channel is congested, the MUGA algorithm is used; otherwise, the Opt. approach is chosen. Then, we further consider the limitation of computation resources on the server and propose an algorithm to assign the tasks of a newly executed application to unoccupied server cores to avoid disturbing the applications already under execution and adjust initial offloading strategy. At last, we reschedule the data transmission if necessary. The proposed method has been tested via simulation experiments, and the results show that our proposed algorithm performs significantly better than previous algorithms.

Chapter 4

Application Level Offline Offloading Strategy

In this chapter, we focus on the application level offloading problem and propose an offline offloading strategy to minimize the average response time of all applications. We study the computation offloading problem in a three-tire offloading system with multiple mobile devices, edge servers and cloud servers, and consider the limitations of the resources of the shared channels, mobile devices and edge servers. In this chapter, we firstly show the system model and formulate the offline offloading problem as a multi-variable non-convex optimization problem. Then, we transform the multi-variable non-convex optimization problem into a single-variable piecewise convex optimization problem. According to the characteristics of transformed problem, we propose an efficient approach based on the binary search to find the optimal offloading strategy. After that, we show that our proposed approach outperforms previous algorithms and the performance of our algorithm is highly robust by experiments. At last, we present the conclusions of our study on application level offloading problem. The findings in this chapter has been published in [59].

4.1 System Model and Formulation

4.1.1 System Model

A three-tier cloud computing system model is considered in this study, as shown in Figure 4.1. For simplicity, we only consider a cloud server and an edge server. The number of mobile devices is denoted as N. The mobile devices are connected to an access point (AP) via a wireless communication network. The wireless communication network includes two shared communication channels: one uplink

channel for data transmissions from the mobile devices to the AP and one downlink channel for data transmissions from the AP to the mobile devices. Since the AP is close to and connected to the edge server via a high-speed link, we assume that the transmission delay between the AP and the edge server can be neglected. The edge server is connected to the cloud server located on the Internet via a wired high-speed link.

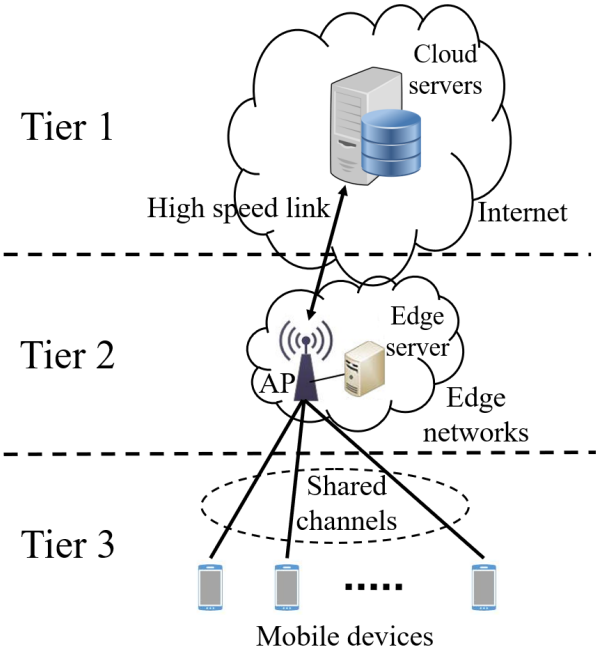


Figure 4.1. The model of a three-tier cloud computing system.

In our system, the average arrival rate of applications at mobile device i is denoted by θ_i. The average service rate of mobile device i is denoted by μ_i. The execution of applications on a mobile device follows the first-come-first-serve (FCFS) basis.

When an application arrives at a mobile device, the mobile device can execute the application locally or offload the application to the edge server. The application is considered to be an inseparable job that should be integrally offloaded, and the offloading ratio of applications that arrive at mobile device i is denoted by α_i. Therefore, the offloading rate of mobile device i is $\alpha_i\theta_i$, and the total offloading rate of all mobile devices is denoted by λ, where $\lambda = \sum_{i=1}^{N} \alpha_i\theta_i$. The edge server can execute an offloaded application or offload the application again to the remote cloud server further. The offloading ratio of the edge server is denoted by α_E, and thus the offloading rate from the edge server to the cloud server is $\alpha_E\lambda$. We assume that the edge server has only one processing server such that it can execute

only one application at a time and execute applications following FCFS basis. On the other hand, the cloud server has abundant processing servers so that each offloaded application can be executed immediately after arriving at the cloud server. The service rates of the edge server and the cloud server are denoted by μ_E and μ_C, respectively.

The network model is shown in Figure 4.2. If an application is decided for offloading, then the input and output data of the application are transmitted via the shared uplink channel from the mobile device to the edge server and via the shared downlink channel from the edge server to the mobile device, respectively. The average data size transmitted over the uplink channel, i.e., the average input data size of the applications, is D_u. The average data size transmitted over the downlink channel, i.e., the average output data size of the applications, is D_d. The transmission rates of the uplink channel and downlink channel are denoted by u and d, respectively, and the total transmission rate of the shared channels is denoted by S ($S = u + d$). Additionally, let μ_u and μ_d denote the average service rates of uplink and downlink channels, respectively, and then we have $\mu_u = \frac{u}{D_u}$ and $\mu_d = \frac{d}{D_d}$. The data transmission via both the uplink and downlink channels also follow the FCFS basis. We assume that the transmission delay between the edge server and the cloud server is a positive constant, denoted by H.

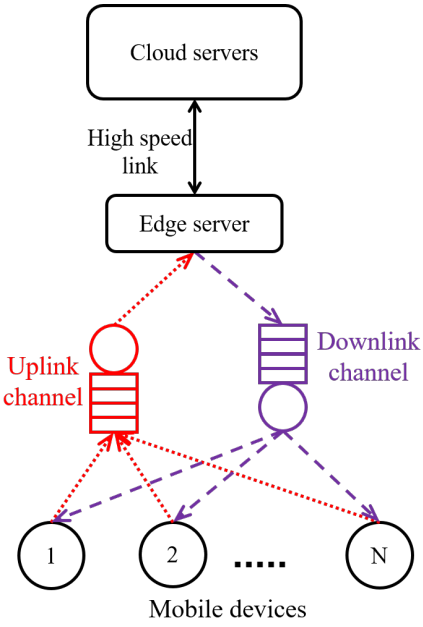


Figure 4.2. Network model.

In order to clearly explain how to execute an application in the system, we show an example in Figure 4.3. When an application arrives at the mobile device

i, it will be offloaded to the edge server with α_i probability. If the application is decided for local execution, it is queued at the mobile device and executed following the FIFO basis. If the application is decided for offloading to the edge server, the mobile device transmits the input data of the application via the uplink channel. The input data transmissions are also queued and transmitted following the FIFO basis. After the input data transmission, the edge server further offloads the applications to the cloud servers with α_E probability. If the application is decided for execution at the edge server, the application is queued at the edge server and executed following the FIFO basis. If the application is decided to be offloaded to the cloud servers, the data transmission and the application execution can be handled immediately because there is no queue at the high-speed link and the cloud servers. After the execution result is obtained by the edge server, the output data are transmitted back to the mobile device via the downlink channel. The output data transmission is also queued and transmitted following the FIFO basis. The response time of the application is the time cost from the application that arrives at the mobile device until the output data are obtained by the mobile device.

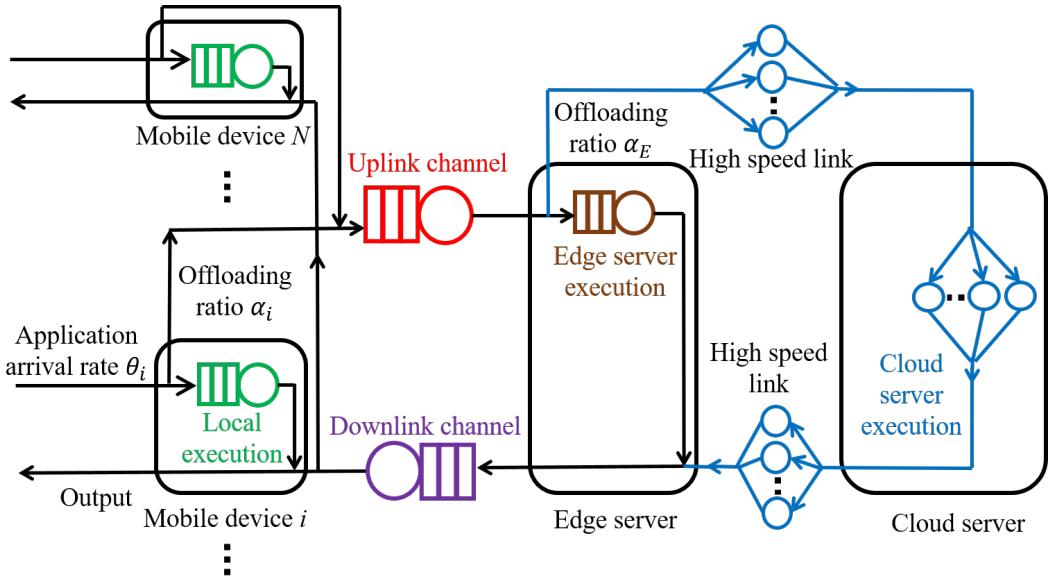


Figure 4.3. An example for the application execution.

In this study, we focus on how to find the optimal offloading ratios for mobile devices and the edge server, i.e., α_i, $i \in \{1, ..., N\}$, and α_E, and how to determine the uplink and downlink transmission rates, i.e., u and d, with limited total transmission rate, respectively, in order to minimize the average response time of applications.

4.1.2 Formulation

The system is divided into five parts as Figure 4.3: the local execution (green part), the uplink channel transmission (red part), the edge server execution (khaki part), the cloud server execution and the data transmission between the edge server and the cloud server (blue part), and the downlink channel transmission (purple part). The notation used in our model and formulation is summarized in Tables 4.1, 4.2 and 4.3.

Table 4.1. Parameters used in application level offloading problem.

Symbol	Meaning
N	number of mobile devices
θ_i	arrival rate of applications at mobile device i
μ_i	service rate of mobile device i
μ_E	service rate of the edge server
μ_C	service rate of the cloud server
D_u	average uplink data size of applications
D_d	average downlink data size of applications
S	total transmission rate of shared communication channels
H	transmission delay of the high-speed link between the edge server and the cloud server
$\overline{h}_i(\alpha_i)$	average execution delay of applications at mobile device i
$h_i(\alpha_i)$	total execution delay of applications at mobile device i
$h_U(\lambda, u)$	total transmission delay of the input data at the uplink channel
$h_D(\lambda, d)$	total transmission delay of the output data at the downlink channel
$h_E(\lambda, \alpha_E)$	total execution delay of applications at the edge server
$h_C(\lambda, \alpha_E)$	sum of total execution time at the cloud server and total transmission time at the high-speed link

Table 4.2. Decision variables used in application level offloading problem.

Symbol	Meaning
u	transmission rate of the uplink channel
d	transmission rate of the downlink channel
α_i	offloading ratio of mobile device i, $i \in \{1, ..., N\}$
α_E	offloading ratio of the edge server

We firstly consider the local execution part. The total execution delay of applications executed at mobile device i is denoted by function $h_i(\alpha_i)$ and the average execution delay of applications is denoted as function $\overline{h}_i(\alpha_i)$, where $1 \geq \alpha_i \geq$

Table 4.3. Intermediate variables used in application level offloading problem.

Symbol	Meaning
λ	total offloading rate of applications from all mobile devices, i.e., $\lambda = \sum_{i=1}^{N} \alpha_i \theta_i$.
μ_u	service rate of the uplink channel, i.e., $\mu_u = \frac{u}{D_u}$
μ_d	service rate of the downlink channel, i.e., $\mu_d = \frac{d}{D_d}$

$\max\{0, \frac{\theta_i - \mu_i}{\theta_i}\}$ to guarantee that the average execution delay at a mobile device is not infinite. The execution delay of the application executed at a mobile device includes the queuing time and the execution time. Then, we can have following equation.

$$h_i(\alpha_i) = (1 - \alpha_i)\theta_i \overline{h}_i(\alpha_i) \tag{4.1}$$

In general, for a nondeterministic queue, i.e., both/one of arrival interval and processing time follow/follows random distributions, the average sojourn time is increasing and convex of its workload and therefore we assume that $\frac{d\overline{h}_i}{d\alpha_i} < 0$ and $\frac{d^2\overline{h}_i}{d{\alpha_i}^2} > 0$. Additionally, we can also have following equations.

$$\frac{dh_i}{d\alpha_i} = (1 - \alpha_i)\theta_i \frac{d\overline{h}_i}{d\alpha_i} - \theta_i \overline{h}_i(\alpha_i) < 0 \tag{4.2}$$

$$\frac{d^2h_i}{d{\alpha_i}^2} = (1 - \alpha_i)\theta_i \frac{d^2\overline{h}_i}{d{\alpha_i}^2} - 2\theta_i \frac{d\overline{h}_i}{d\alpha_i} > 0 \tag{4.3}$$

Thus, we know that h_i is a decreasing convex function, i.e., $\frac{dh_i}{d\alpha_i} \leq 0 \leq \frac{d^2h_i}{d{\alpha_i}^2}$. According to Kingman's formula [60], we know that for a single processor nondeterministic queue, the average queuing time becomes infinitely large if the arrival rate reaches the processing rate. Thus, we let $h_i(\alpha_i) = +\infty$ when $\theta_i \geq \mu_i$ and $(1 - \alpha_i)\theta_i = \mu_i$.

Let us consider the uplink channel transmission part. The total transmission delay of the input data of offloaded applications is denoted by function $h_U(\lambda, u)$, where $\lambda \leq \mu_u$. The transmission delay of the data transmitted via the uplink channel includes the queuing time and the transmission time. Similar to h_i, we assume that h_U is an increasing convex function of λ, i.e., $\frac{\partial h_U}{\partial \lambda} \geq 0$ and $\frac{\partial^2 h_U}{\partial \lambda^2} \geq 0$, while a decreasing convex function of u, i.e., $\frac{\partial h_U}{\partial u} \leq 0$, and $\frac{\partial^2 h_U}{\partial u^2} \geq 0$. Additionally, we let $h_U(\lambda, u) = +\infty$ when $\lambda = \mu_u$.

At the edge server execution part, the total execution delay of applications at the edge server is denoted by function $h_E(\lambda, \alpha_E)$, where $(1 - \alpha_E)\lambda \leq \mu_E$. The

execution delay of an application executed at the edge server includes the queuing time and the execution time. Similar to h_i, we assume that h_E is an increasing convex function of λ, i.e., $\frac{\partial h_E}{\partial \lambda} \geq 0$ and $\frac{\partial^2 h_E}{\partial \lambda^2} \geq 0$, while a decreasing convex function of α_E, i.e., $\frac{\partial h_E}{\partial \alpha_E} \leq 0$ and $\frac{\partial^2 h_E}{\partial {\alpha_E}^2} \geq 0$. Additionally, we have $h_E(\lambda, \alpha_E) = +\infty$ when $(1 - \alpha_E)\lambda = \mu_E$.

After that, we consider the cloud server execution and the data transmission between the edge server and the cloud server. The total execution delay of application at the cloud server equals to the sum of the total execution time at the cloud server and the total transmission time over the communication channel and is denoted by function $h_C(\lambda, \alpha_E) = (2H + \frac{1}{\mu_C})\alpha_E \lambda$. Note that there is neither queue on the high-speed channel nor on the cloud server, only the execution time and the transmission time are included in h_C. It is clear that h_C is an increasing convex function of λ and α_E when $\lambda \geq 0$ and $\alpha_E \geq 0$.

At last, we consider the downlink transmission part. The total transmission delay of the output data of offloaded applications is denoted by function $h_D(\lambda, d)$, where $\lambda \leq \mu_d$. The transmission delay of the data transmitted via the downlink channel includes the queuing time and the transmission time. Similar to h_U, we assume that $\frac{\partial h_D}{\partial \lambda} \geq 0$, $\frac{\partial h_D}{\partial d} \leq 0$, $\frac{\partial^2 h_D}{\partial \lambda^2} \geq 0$ and $\frac{\partial^2 h_D}{\partial d^2} \geq 0$. Additionally, we have $h_D(\lambda, d) = +\infty$ when $\lambda = \mu_d$.

In summary, the problem of minimizing the average response time of applications considered in this chapter is formulated as follows:

$$\min \quad \frac{1}{\sum_{i=1}^{N} \theta_i} \left(\sum_{i=1}^{N} h_i(\alpha_i) + h_U(\lambda, u) + h_E(\lambda, \alpha_E) + h_C(\lambda, \alpha_E) + h_D(\lambda, d) \right), \tag{4.4}$$

subject to

$$\lambda = \sum_{i=1}^{N} \alpha_i \theta_i, \tag{4.5}$$

$$(1 - \alpha_i)\theta_i \leq \mu_i, \quad i \in \{1, ..., N\}, \tag{4.6}$$

$$\lambda \geq \sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}, \tag{4.7}$$

$$\lambda \leq \sum_{i=1}^{N} \theta_i, \tag{4.8}$$

$$S = u + d, \tag{4.9}$$

$$\lambda \leq \mu_u, \tag{4.10}$$

$$\lambda \leq \mu_d, \tag{4.11}$$

$$(D_u + D_d)\lambda \leq S, \tag{4.12}$$
$$(1 - \alpha_E)\lambda \leq \mu_E, \tag{4.13}$$
$$0 \leq \alpha_i \leq 1, \quad i \in \{1, ..., N\}, \tag{4.14}$$
$$0 \leq \alpha_E \leq 1. \tag{4.15}$$

Constraint (4.5) ensures that the total offloading rate is the sum of offloading rates of all mobile devices. Constraints (4.6) and (4.7) guarantee that the execution delay at a mobile device is not infinite. Constraint (4.8) guarantees that the total offloading rate of mobile devices is lower than total arrival rate of mobile devices. Constraint (4.9) shows that the total transmission rate is the sum of the uplink transmission rate and the downlink transmission rate. Constraints (4.10) and (4.11) guarantee that the transmission delays at the uplink and downlink channels are not infinite. Constraint (4.12) guarantees that the minimum transmission rate requirement is lower than the total transmission rate. Constraint (4.13) guarantees that the execution delay at the edge server is not infinite. Constraints (4.14) and (4.15) state the range of offloading ratios of mobile devices and the edge server.

4.2 Analysis and Algorithm for Application Level Offloading

In this section, we analyse the computation offloading system and further propose an efficient algorithm to determine the optimal transmission rate of uplink and downlink channels, and the optimal offloading strategy for mobile devices and the edge server. Since θ_i are given as parameters, $\frac{1}{\sum_{i=1}^{N} \theta_i}$ in problem (4.4) can be removed and the average response time minimizing problem becomes the total response time minimizing problem. The system is divided into three part in our analysis: the local execution part including the application executions on mobile devices, the data transmission part including the data transmissions over the shared wireless communication channels and the remote execution part including the application executions on servers and data transmission between the edge server and the cloud server. Then, the problem is transformed into a piecewise convex optimization problem based on our analysis and an algorithm is proposed to solve the transformed problem.

4.2.1 Data Transmission over Uplink and Downlink Channels

We firstly consider how to determine the uplink and downlink transmission rates u and d to minimize the total transmission delay for a given total offloading rate

λ. We simply write $h_U(\lambda, u)$ and $h_D(\lambda, d)$ as $h_U(u)$ and $h_D(d)$, respectively, with a given λ. The total transmission delay, denoted by f_1, is given as follows:

$$f_1 = h_U(u) + h_D(d), \tag{4.16}$$

where

$$S = u + d, \tag{4.17}$$
$$\lambda \leq \mu_u, \tag{4.18}$$
$$\lambda \leq \mu_d. \tag{4.19}$$

Theorem 2 *For a given λ, the minimum total transmission delay, denoted as ${f_1}^*$, and the optimal transmission rates of uplink and downlink channels, denoted as u^* and d^*, can be uniquely determined if constraint (4.12) is held. Additionally, ${f_1}^*$ is a convex function of λ.*

Proof *The transmission rate determination problem is formulated as follows.*

$$\begin{aligned} \min \quad & f_1 = h_U(u) + h_D(d), && (4.20)\\ \textit{subject to} & \\ & S = u + d, && (4.21)\\ & \lambda \leq \mu_u, && (4.22)\\ & \lambda \leq \mu_d. && (4.23) \end{aligned}$$

We firstly analyze the feasible region of problem (4.20). If $(D_u + D_d)\lambda > S$, at least one of constraints (4.22) and (4.23) cannot be held because $S = u + d$ (constraint (4.21)), $\mu_u = \frac{u}{D_u}$ and $\mu_d = \frac{d}{D_d}$. Thus, constraint (4.12) should be held to guarantee that the optimal solution of problem (4.20) exists.

Then, we need to find the optimal solution of problem (4.20). The optimal transmission rates cannot be determined when $\lambda = \mu_u$, $\lambda < \mu_d$ or $\lambda < \mu_u$, $\lambda = \mu_d$ because $h_U(u) = +\infty$ when $\lambda = \mu_u$ and $h_D(d) = +\infty$ when $\lambda = \mu_d$. As described in section 4.1.2, h_U and h_D are convex functions of u and d, respectively. Thus, problem (4.20) is a convex optimization problem in terms of u and d for a given λ. A lagrange multiplier [62] k_1 of constraint (4.21) is introduced to analyze the problem. Thus, problem (4.20) is transformed into problem (4.24).

$$\min \quad L_1 = f_1 + k_1(S - u - d). \tag{4.24}$$

Then we have $\frac{\partial L_1}{\partial u} = 0$ and $\frac{\partial L_1}{\partial d} = 0$ when L_1 is minimized. Thus, we have Equation (4.25).

$$\frac{\partial f_1(u^*)}{\partial u} = \frac{\partial f_1(d^*)}{\partial d} = k_1. \tag{4.25}$$

According to constraints (4.21)–(4.23) and Equation (4.25), the optimal u^* and d^* are obtained by solving following relations.

$$S = u^* + d^*, \tag{4.26}$$

$$\frac{\partial h_U(u^*)}{\partial u} = \frac{\partial h_D(d^*)}{\partial d}, \tag{4.27}$$

$$D_u\lambda \leq u^*, \tag{4.28}$$

$$D_d\lambda \leq d^*. \tag{4.29}$$

The minimum total transmission delay is obtained by substituting u^* and d^* into problem (4.20) and shown as follows.

$$f_1{}^* = h_U(u^*) + h_D(d^*). \tag{4.30}$$

Note that both u^* and d^* depend on λ. Let $u^* = g_U(\lambda)$ and $d^* = g_D(\lambda)$. According to Equations (4.26) and (4.27), we know that $\frac{dg_U(\lambda)}{d\lambda} + \frac{dg_D(\lambda)}{d\lambda} = 0$ and $\frac{\partial h_U}{\partial g_U(\lambda)} = \frac{\partial h_D}{\partial g_D(\lambda)}$. We can then find $\frac{\partial^2 f_1{}^*}{\partial \lambda^2} \geq 0$ by using the chain rule [61]. ■

Once u^* and d^* have been found, the constraints about u and d, i.e., constraints (4.9)–(4.11), can be neglected.

4.2.2 Remote Execution

We secondly consider how to determine the offloading ratio of the edge server, i.e., α_E, for a given λ. We simply write $h_E(\lambda, \alpha_E)$ and $h_C(\lambda, \alpha_E)$ as $h_E(\alpha_E)$ and $h_C(\alpha_E)$, respectively, with a given λ. The total execution delay of the offloaded applications at servers, denoted by f_2, is given as follows:

$$f_2 = h_E(\alpha_E) + h_C(\alpha_E), \tag{4.31}$$

where

$$(1 - \alpha_E)\lambda \leq \mu_E, \tag{4.32}$$

$$0 \leq \alpha_E \leq 1. \tag{4.33}$$

Theorem 3 *For a given λ, the minimum total execution delay, denoted as ${f_2}^*$, of the offloaded applications at servers and the optimal offloading ratio of the edge server, denoted as ${\alpha_E}^*$, can be uniquely determined. Additionally, ${f_2}^*$ is a convex function of λ.*

Proof *The edge server offloading ratio determination problem is formulated as follows.*

$$\min \qquad f_2 = h_E(\alpha_E) + h_C(\alpha_E), \tag{4.34}$$

subject to

$$(1-\alpha_E)\lambda \leq \mu_E, \tag{4.35}$$

$$0 \leq \alpha_E \leq 1. \tag{4.36}$$

As described in section 4.1.2, h_E *and* h_C *are convex functions of* α_E *for a given* λ. *Thus,* $h_E + h_C$ *is a convex function of* α_E. *By taking the derivatives of* h_E *and* h_C, *the optimal offloading ratio, i.e.,* ${\alpha_E}^*$, *can be found. If* $\frac{\partial h_E(0)}{\partial \alpha_E} + \frac{\partial h_C(0)}{\partial \alpha_E} \geq 0$, ${\alpha_E}^* = 0$; *if* $\frac{\partial h_E(0)}{\partial \alpha_E} + \frac{\partial h_C(0)}{\partial \alpha_E} < 0 < \frac{\partial h_E(1)}{\partial \alpha_E} + \frac{\partial h_C(1)}{\partial \alpha_E}$, $1 < {\alpha_E}^* < 1$; *and if* $\frac{\partial h_E(1)}{\partial \alpha_E} + \frac{\partial h_C(1)}{\partial \alpha_E} \leq 0$, ${\alpha_E}^* = 1$. *Therefore, we obtain* ${\alpha_E}^*$ *as follows, where* $G(\alpha_E) = \frac{\partial h_E}{\partial \alpha_E} + \frac{\partial h_C}{\partial \alpha_E}$ *and* G^{-1} *is the inverse function of* G.

$$\alpha_E^* = \begin{cases} 0, & \text{if } \frac{\partial h_E(0)}{\partial \alpha_E} + \frac{\partial h_C(0)}{\partial \alpha_E} \geq 0, \\ 1, & \text{if } \frac{\partial h_E(1)}{\partial \alpha_E} + \frac{\partial h_C(1)}{\partial \alpha_E} \leq 0, \\ G^{-1}(0), & \text{otherwise.} \end{cases} \tag{4.37}$$

Let ${\alpha_E}^* = g_E(\lambda)$, *the minimum total execution delay of offloaded applications is obtained by substituting* $g_E(\lambda)$ *into problem (4.34) and shown as follows.*

$${f_2}^* = h_E(g_E(\lambda)) + h_C(g_E(\lambda)). \tag{4.38}$$

According to Equations (4.37), we find that $\frac{\partial h_E}{\partial g_E(\lambda)} + \frac{\partial h_C}{\partial g_E(\lambda)} = 0$. *We also have* $\frac{\partial h_C}{\partial g_E(\lambda)} = \lambda(2H + \frac{W}{C})$. *Therefore, we find* $\frac{\partial^2 {f_2}^*}{\partial \lambda^2} \geq 0$ *by using the chain rule.* ■

Once ${\alpha_E}^*$ has been found, the constraints about α_E, i.e., constraints (4.13) and (4.15), can be neglected. From Theorems 2 and 3, problem (4.4) can be transformed into the following optimization problem, where constraints (4.9)–(4.11), (4.13) and (4.15) are neglected.

$$\min \qquad F_1 = \sum_{i=1}^{N} h_i(\alpha_i) + {f_1}^*(\lambda) + {f_2}^*(\lambda), \tag{4.39}$$

subject to

$$\lambda = \sum_{i=1}^{N} \alpha_i \theta_i, \tag{4.40}$$

$$(1-\alpha_i)\theta_i \leq \mu_i, \quad i \in \{1, ..., N\}, \tag{4.41}$$

$$\lambda \geq \sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}, \tag{4.42}$$

$$\lambda \leq \sum_{i=1}^{N} \theta_i, \tag{4.43}$$

$$(D_u + D_d)\lambda \leq S, \tag{4.44}$$

$$0 \leq \alpha_i \leq 1, \quad i \in \{1, ..., N\}. \tag{4.45}$$

4.2.3 Local Execution

At last, we consider how to determine the offloading rates of mobile devices for a given λ to minimize the total execution delay of mobile devices. Let λ_i denote the offloading rate of mobile device i; then, we have $\lambda_i = \alpha_i \theta_i$, $i \in \{1, 2, ..., N\}$ and $\lambda = \sum_{i=1}^{N} \lambda_i$. Since $\alpha_i = \frac{\lambda_i}{\theta_i}$ is an affine function and $h_i(\alpha_i)$ is a convex function, $h_i(\frac{\lambda_i}{\theta_i})$ is a convex function of λ_i. Because θ_i is a constant, henceforth, we use the simple function $H_i(\lambda_i)$ rather than $h_i(\frac{\lambda_i}{\theta_i})$ for description simplicity. The total execution delay of all mobile devices, denoted as f_3, is given as follows:

$$f_3 = \sum_{i=1}^{N} H_i(\lambda_i), \tag{4.46}$$

where

$$\lambda = \sum_{i=1}^{N} \lambda_i, \tag{4.47}$$

$$\theta_i - \lambda_i \leq \mu_i, \quad i \in \{1, ..., N\}, \tag{4.48}$$

$$0 \leq \lambda_i \leq \theta_i, \quad i \in \{1, ..., N\}. \tag{4.49}$$

An example is used to make our main idea for solving offloading rate determination problem with a given λ clearly. As shown in Figure 4.4, there are four reservoirs with different depths, widths and heights. The volumes of water in each reservoir are also different (the water level may be higher than the height of the reservoir such as reservoir A in Figure 4.4). Let us consider that how to pump water from reservoirs while minimizing the maximal absolute water level heights in reservoirs.

In Figure 4.4, reservoir A is different with others so that we classify it as a separate class and firstly pump water from reservoir A until the water level in A is same as the height of A (Figure 4.5(a)). Then, we continue pumping water from reservoir A until the absolute water level in A is same as that in B (Figure 4.5(b)). Next, we pump water from both of reservoirs A and B until the absolute water levels in A and B are same as that in C (Figure 4.5(c)). After that, the water is pumped from reservoirs A, B and C until the absolute water levels in A, B and

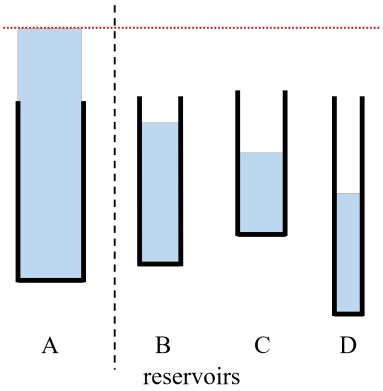


Figure 4.4. Original state.

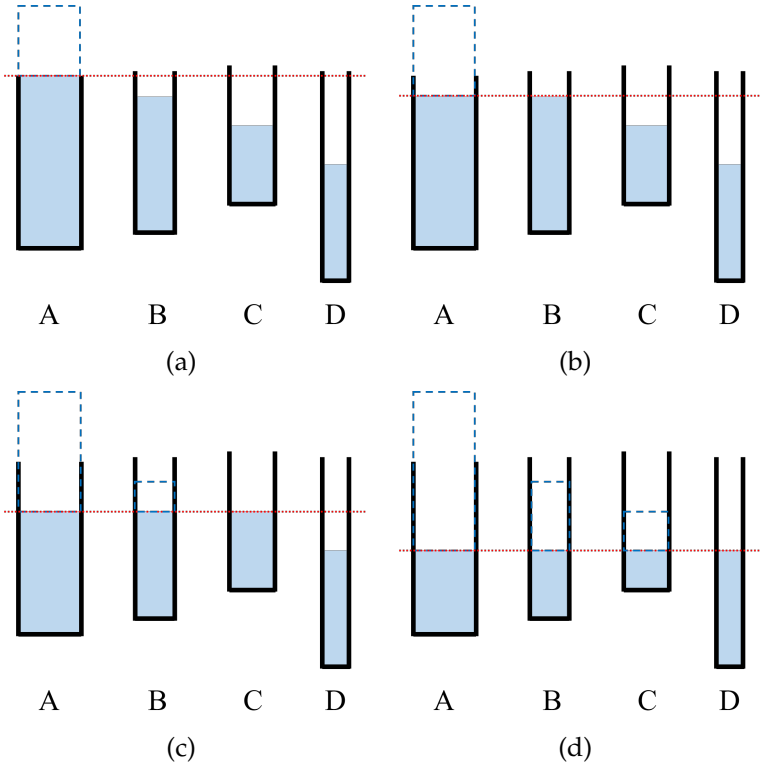


Figure 4.5. Intermediate states 1.

C are same as that in D (Figure 4.5(d)). When the reservoirs A, B, C and D become the same absolute water level, all of reservoirs need to be pumped until one of them becomes empty (C in Figure 4.6(a)). The empty reservoir C is classified as a separate class (Figure 4.6(b)) because there is not water can be pumped from C. Similarly, if we continue pumping from reservoirs, B will become empty and be

classified with C (Figure 4.6(c)). Then, A becomes empty and is classified with B and C (Figure 4.6(c)). At last, all of reservoirs become empty and the water is

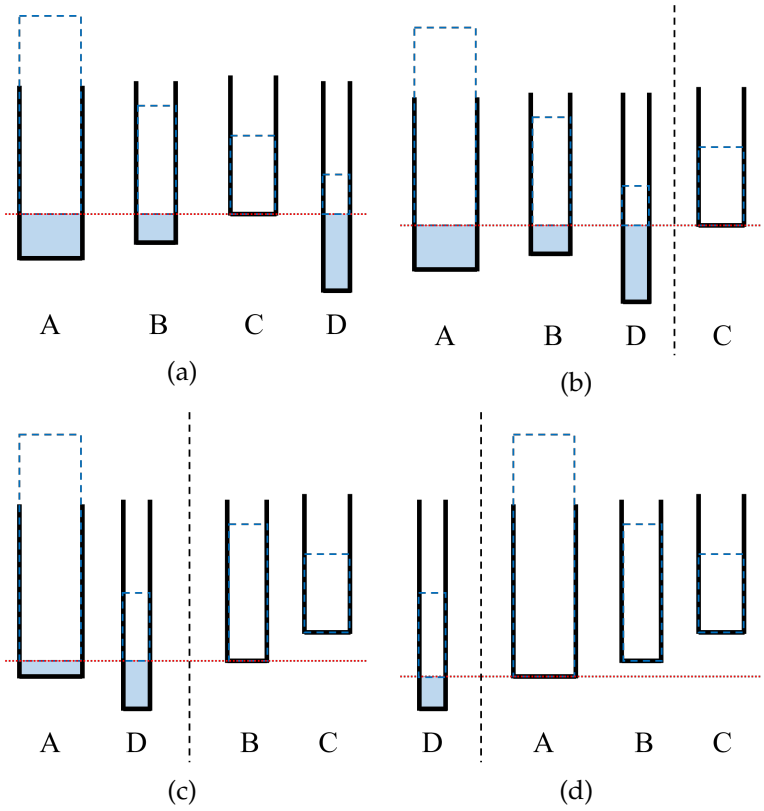


Figure 4.6. Intermediate states 2.

pumped from reservoirs completely (Figure 4.7).

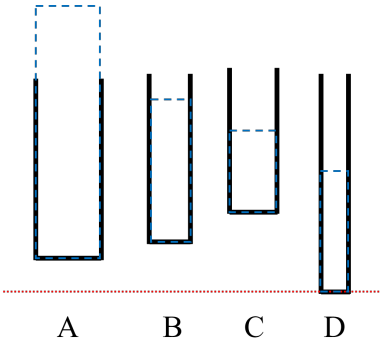


Figure 4.7. Final state.

In our problem, each mobile device can be treated as a reservoir and the application offloading can be seen as the water pumping. Different processing rates of mobile devices also correspond to different depths, widths and heights of reservoirs. Additionally, gradients of offloading gains for mobile devices can be treated as the water levels of reservoirs. Then, we can prove following theorem.

Theorem 4 *For a given λ, the minimum total execution delay, denoted as ${f_3}^*$, can be uniquely determined if constraints (4.42) and (4.43) are held. The minimum total execution delay and the optimal offloading rate of mobile device i, denoted as ${\lambda_i}^*$, are shown in Equations (4.50) and (4.51), where g_i is a function of λ. Additionally, ${f_3}^*$ is a piecewise convex function of λ that is divided by $2N - k$ nondifferentiable points where k is the number of overloaded mobile without offloading, i.e., $k = |\mathcal{N}_{OL}|$ and $\mathcal{N}_{OL} = \{i|\mu_i \leq \theta_i,\ i \in \{1, ..., N\}\}$.*

$$ {f_3}^* = \sum_{i=1}^{N} H_i({\lambda_i}^*), \tag{4.50} $$

$$ {\lambda_i}^* = g_i(\lambda), \quad i \in \{1, ..., N\}. \tag{4.51} $$

Proof *We formulate the offloading rate determination problem as follows.*

$$ \min \qquad f_3 = \sum_{i=1}^{N} H_i(\lambda_i), \tag{4.52} $$

subject to

$$ \lambda = \sum_{i=1}^{N} \lambda_i, \tag{4.53} $$

$$ \theta_i - \lambda_i \leq \mu_i, \quad i \in \{1, ..., N\}, \tag{4.54} $$

$$ 0 \leq \lambda_i \leq \theta_i, \quad i \in \{1, ..., N\}. \tag{4.55} $$

Similar to the proof of Theorem 2, constraints (4.42) and (4.43) should be held to guarantee that the optimal solution of problem (4.52) exists. Additionally, if mobile device i is overloaded, i.e., $\theta_i \geq \mu_i$, the optimal offloading rates cannot be determined for when $\lambda_i = \theta_i - \mu_i$ because $H_i(\theta_i - \mu_i) = +\infty$.

Since H_i is a convex function of λ_i, which is described above, f_3 is a convex function of λ_i, $i \in \{1, ..., N\}$. Thus, f_3 has only one extreme value in the feasible region of problem (4.52).

The mobile devices whose optimal offloading rates are determined to lie on the lower bound are represented by the set $\mathcal{N}_{full}$, i.e., $i \in \mathcal{N}_{full}$ if the optimal offloading rate of mobile device i, ${\lambda_i}^*$, is 0, which means that no application is offloaded from mobile device i. The mobile devices whose optimal offloading rates are determined to lie on the upper bound are represented by the set $\mathcal{N}_{idle}$, i.e., $i \in \mathcal{N}_{idle}$ if ${\lambda_i}^* = \theta_i$, which means that all

applications are offloaded from mobile device i. All other mobile devices are represented by the set $\mathcal{N}_{normal}$, i.e., $i \in \mathcal{N}_{normal}$ if $0 < {\lambda_i}^* < \theta_i$, which means that some applications are offloaded from mobile device i. According to KKT conditions [63], the optimal offloading rates, i.e., the optimal solution of problem (4.52), have to satisfy following relations for a given λ, where k_2 is a lagrange multiplier.

$$\frac{dH_i({\lambda_i}^*)}{d\lambda_i} = k_2, \quad i \in \mathcal{N}_{normal}, \tag{4.56}$$

$$\frac{dH_i(0)}{d\lambda_i} \geq k_2, \quad i \in \mathcal{N}_{full}, \tag{4.57}$$

$$\frac{dH_i(\theta_i)}{d\lambda_i} \leq k_2, \quad i \in \mathcal{N}_{idle}, \tag{4.58}$$

$$\mu_i > \theta_i, \quad i \in \mathcal{N}_{full}, \tag{4.59}$$

$$\lambda = \sum_{i \in \mathcal{N}_{normal}} {\lambda_i}^* + \sum_{i \in \mathcal{N}_{idle}} \theta_i. \tag{4.60}$$

For a given λ, the optimal offloading rates of mobile devices are found by solving relations (4.56)–(4.60). We express the optimal offloading rate of mobile device i as a function of λ, i.e., ${\lambda_i}^* = g_i(\lambda)$, and thus the optimal total execution delay of mobile devices, i.e., ${f_3}^*$ is a function of λ. Note that, function ${f_3}^*$ has a set of nondifferentiable points. For example

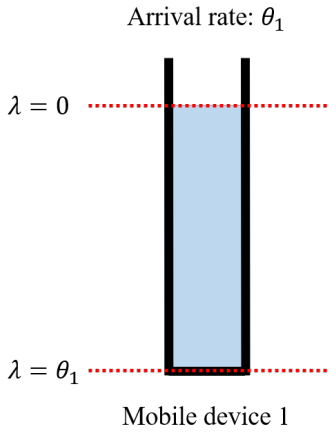


Figure 4.8. Nondifferentiable points for one mobile device.

there is only one mobile device as Figure 4.8, a nondifferentiable point of ${f_3}^*$ exists if $\lambda = 0$ or $\lambda = \theta_1$. If there are two mobile devices as Figure 4.9, a nondifferentiable point of ${f_3}^*$ exists if $\lambda = 0$, $\lambda = \theta_1 - {G_1}^{-1}(\frac{dH_2(0)}{d\lambda_2})$, $\lambda = \theta_1 + {G_2}^{-1}(\frac{dH_1(\theta_1)}{d\lambda_1})$ or $\lambda = \theta_1 + \theta_2$, where $G_1 = \frac{dH_1(\lambda_1)}{d\lambda_1}$ and $G_2 = \frac{dH_2(\lambda_2)}{d\lambda_2}$. Therefore, ${f_3}^*$ is a piecewise function of λ and has $2N$ nondifferentiable points if there are N mobile devices. Additionally, since constraint (4.42)

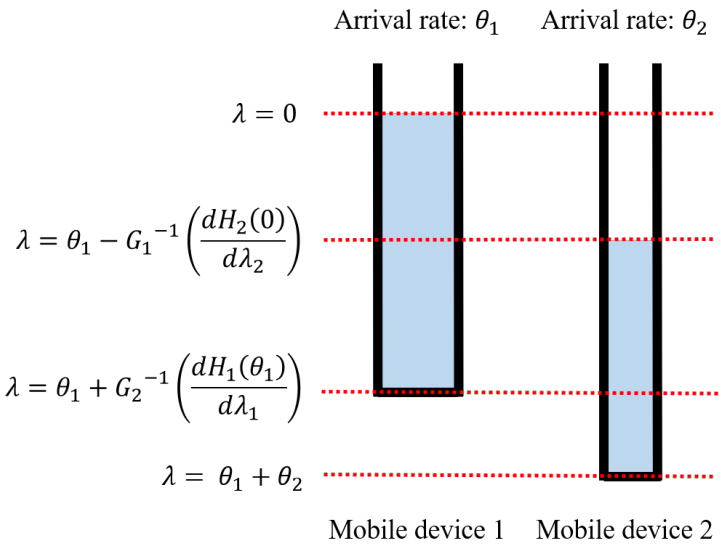


Figure 4.9. Nondifferentiable points for two mobile devices.

is held and $\mathcal{N}_{normal} = \{i|\mu_i \leq \theta_i, i \in \{1,...,N\}\}$ *when* $\lambda = \sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}$, *the number of nondifferentiable points of* ${f_3}^*$ *in its feasible region is* $2N - k$ *where* $k = |\mathcal{N}_{OL}|$ *and* $\mathcal{N}_{OL} = \{i|\mu_i \leq \theta_i,\ i \in \{1,...,N\}\}$. *Moreover, we find that* ${f_3}^*$ *is a convex function of* λ *in each piece by using the chain rule. Thus,* ${f_3}^*$ *is a convex piecewise function of* λ.

Once the optimal offloading rate for each mobile device has been found, constraints (4.40), (4.41) and (4.45) can be neglected. Problem (4.39) can be further transformed into the following optimization problem, where constraints (4.42)–(4.44) are consolidated into constraint (4.62).

$$\min \qquad F_2 = {f_1}^*(\lambda) + {f_2}^*(\lambda) + {f_3}^*(\lambda), \tag{4.61}$$

subject to

$$\sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\} \leq \lambda \leq \min\{\sum_{i=1}^{N} \theta_i, \frac{S}{D_u + D_d}\}. \tag{4.62}$$

4.2.4 Algorithm for Application Level Offloading

As described in Theorem 4, ${f_3}^*(\lambda)$ is a piecewise convex function of λ divided by $2N - k$ nondifferentiable points. Thus, problem (4.61) can be decomposed into $2N - k$ convex subproblems, $A_j (j \in \{k+1,...,2N\})$, and shown as follows:

$$\min \quad F_2 = \min\{A_{k+1},...,A_{2N}\}. \tag{4.63}$$

Each subproblem A_j, $j \in \{k+1, ..., 2N\}$, is formulated as follows:

$$\min \qquad A_j = {f_1}^*(\lambda) + {f_2}^*(\lambda) + {f_3}^*(\lambda), \tag{4.64}$$

subject to

$$\varphi_{j-1} \leq \lambda \leq \varphi_j, \tag{4.65}$$

$$\lambda \leq \frac{S}{D_u + D_d}, \tag{4.66}$$

where φ_j, $j \in \{k+1, ..., 2N\}$, are values of nondifferentiable points of ${f_3}^*(\lambda)$,which are sorted in ascending order, and $\varphi_k = \sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}$. Because $\sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}$ is the lower bound of λ in problem (4.61) and φ_j, $j \in \{k+1, ..., 2N\}$, are located in the feasible region of problem (4.61), we have $\varphi_j \geq \varphi_k = \sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}$, $j \in \{k+1, ..., 2N\}$. Additionally, as described in the proof of Theorem 4, $\sum_{i=1}^{N} \theta_i$ is the largest value among nondifferentiable points, i.e., $\varphi_j \leq \sum_{i=1}^{N} \theta_i$, $j \in \{k+1, ..., 2N\}$. Therefore, constraint (4.62) can be simplified to constraint (4.66).

Consequently, in order to solve problem (4.64), we need to find φ_j, $j \in \{k+1, ..., 2N\}$. First, we calculate $\frac{dH_i(0)}{d\lambda_i}$ and $\frac{dH_i(\theta_i)}{d\lambda_i}$, $i \in \{1, ..., N\}$. Note that if mobile device i is overloaded without offloading, i.e., $\theta_i \geq \mu_i$, $\lambda_i = 0$ is not in the feasible region of H_i. Therefore, for an overloaded mobile device i, we let $\frac{dH_i(0)}{d\lambda_i} = -\infty$. Second, we set $\mathcal{N} = \{\frac{dH_i(0)}{d\lambda_i}, \frac{dH_i(\theta_i)}{d\lambda_i} | i \in \{1, ..., N\}\}$ and sort $\mathcal{N}$ in ascending order. Then, φ_j, $j \in \{k+1, ..., N\}$, corresponds jth element of $\mathcal{N}$. Note that $\{\varphi_j | j \in \{k+1, ..., 2N\}\}$ is also the set of the first nondifferentiable points of $g_i(\lambda)$, $i \in \{1, ..., N\}$, if $\theta_i < \mu_i$, and the last nondifferentiable points of $g_i(\lambda)$, $i \in \{1, ..., N\}$, as described in the proof of Theorem 4. As λ increases, the first nondifferentiable point of $g_i(\lambda)$ is the point at which the applications of mobile device i are beginning to be offloaded and the last nondifferentiable point of $g_i(\lambda)$ is the point at which all the applications of this mobile device are offloaded. Therefore, if the jth ($j \geq k+1$) element of $\mathcal{N}$ is $\frac{dH_i(0)}{d\lambda_i}$, φ_j is obtained by solving following relations.

$$\frac{dg_i({\varphi_j}^-)}{d\lambda} = 0, \tag{4.67}$$

$$\frac{dg_i({\varphi_j}^+)}{d\lambda} < 0, \tag{4.68}$$

where $\frac{dg_i({\varphi_j}^-)}{d\lambda}$ and $\frac{dg_i({\varphi_j}^+)}{d\lambda}$ are the left derivative and right derivative, respectively, of $g_i(\lambda)$ at φ_j. If the jth ($j \geq k+1$) element of $\mathcal{N}$ is $\frac{dH_i(\theta_i)}{d\lambda_i}$, φ_j is obtained by solving

following relations.

$$\frac{dg_i(\varphi_j{}^-)}{d\lambda} < 0, \tag{4.69}$$

$$\frac{dg_i(\varphi_j{}^+)}{d\lambda} = 0. \tag{4.70}$$

After finding the nondifferentiable points, we need to find in which piece the optimal total offloading rate, denoted by λ^*, is located. Let φ denote a nondifferentiable point of F_2. If $\frac{dF_2(\varphi^-)}{d\lambda} \geq 0$, which means that we can further offload applications from mobile devices to servers, then we must have $\frac{dF_2(\varphi^-)}{d\lambda} \geq 0 \Leftrightarrow \frac{dF_2(\varphi^+)}{d\lambda} \geq 0$. For the same reason, we have $\frac{dF_2(\varphi^-)}{d\lambda} \leq 0 \Leftrightarrow \frac{dF_2(\varphi^+)}{d\lambda} \leq 0$. Therefore, the optimal piece, denoted by j^*, satisfies the conditions $\varphi_{j^*-1} \leq \lambda^* \leq \varphi_{j^*}$ and $\frac{dA_{j^*}(\varphi_{j^*-1}{}^+)}{d\lambda} \leq 0 \leq \frac{dA_{j^*}(\min\{\varphi_{j^*}, \frac{S}{Du+D_d}\}^-)}{d\lambda}$. A nonoptimal piece j satisfies either the conditions of $\frac{dA_j(\varphi_{j-1}{}^+)}{d\lambda} \leq 0$ and $\frac{dA_j(\min\{\varphi_j, \frac{S}{Du+D_d}\}^-)}{d\lambda} \leq 0$ or the conditions of $\frac{dA_j(\varphi_{j-1}{}^+)}{d\lambda} \geq 0$ and $\frac{dA_j(\min\{\varphi_j, \frac{S}{Du+D_d}\}^-)}{d\lambda} \geq 0$. Thus, j^* can be easily found using the binary search method [64]. Additionally, if any equality holds in the above conditions, λ^* can be found directly, i.e., $\lambda^* = \varphi_{j-1}$ if $\frac{dA_j(\varphi_{j-1}{}^+)}{d\lambda} = 0$ or $\lambda^* = \varphi_j$ if $\frac{dA_j(\min\{\varphi_j, \frac{S}{Du+D_d}\}^-)}{d\lambda} = 0$. If no equality holds, λ^* can be found using Newton's method [65] in the optimal piece. The optimal offloading rates of the mobile devices, denoted as ${\alpha_i}^*$, $i \in \{1, ..., N\}$, are obtained by letting ${\alpha_i}^* = \frac{g_i(\lambda^*)}{\theta_i}$.

Our proposed offloading algorithm is shown in Algorithm 8. Nondifferentiable points are found in steps $1 \sim 11$. j^* and λ^* are found in steps 12 and 13. In step 14, the optimal offloading ratios are calculated and the transmission rates are determined. The computational complexity of step 10 is bounded by $O(N \log N)$, where N is the number of mobile devices. The computational complexity of binary search method used in step 12 is bounded by $O(\log N)$. The computational complexity of Newton's method used in step 13 is bounded by $O(1)$. Therefore, the total computational complexity of Algorithm 8 is bounded by $O(N \log N)$.

4.3 Performance Evaluation

We present the performance evaluation of our proposed efficient offloading algorithm, denoted by *ALO* in the figures. An extreme case in which no application is offloaded, called *No offloaded*, is also simulated for comparison. Two previous algorithms [33, 34], denoted by *IPM-based* and *ERWP*, are compared with our proposed algorithm. The algorithm proposed in [33] considered the queue on the

Algorithm 8 Application level offloading algorithm.

Input: $\theta_i, \mu_i,\ i \in \{1, ..., N\}$, D_u, D_d, S, μ_E, μ_C, H

set $k = 0, \mathcal{N} = \emptyset$
for all $i \in \{1, ..., N\}$ **do**
  if $\theta_i \geq \mu_i$ **then**
    set $\frac{dH_i(0)}{d\lambda_i} = -\infty$, $k = k + 1$, calculate $\frac{dH_i(\theta_i)}{d\lambda_i}$
  else
    calculate $\frac{dH_i(0)}{d\lambda_i}$ and $\frac{dH_i(\theta_i)}{d\lambda_i}$
  end if
  set $\mathcal{N} = \mathcal{N} \cup \{\frac{dH_i(0)}{d\lambda_i}, \frac{dH_i(\theta_i)}{d\lambda_i}\}$
end for
sort $\mathcal{N}$ in ascending order
obtain $\varphi_j,\ j \in \{k + 1, ..., 2N\}$, by solving either relations (4.67) and (4.68) or (4.69) and (4.70); set $\varphi_k = \sum_{i=1}^{N} \max\{0, \theta_i - \mu_i\}$
find j^* using the binary search method
find λ^* using Newton's method in A_{i^*}
calculate u^*, d^*, ${\alpha_E}^*$ and ${\alpha_i}^*,\ i \in \{1, ..., N\}$, using relations (4.26)–(4.29), (4.37) and ${\alpha_i}^* = \frac{g_i(\lambda^*)}{\theta_i}$, respectively
return u^*, d^*, ${\alpha_E}^*$ and ${\alpha_i}^*,\ i \in \{1, ..., N\}$

edge server but did not consider the queue on the shared channels and the cooperation between servers. The algorithm proposed in [34] considered the queue on the channels but did not consider the queue on the edge server. Because both previous algorithms treated the service rate of a mobile device as a constant value, we assume that the service rates of all mobile devices are the same in our experiments, i.e., $\mu_i = \mu,\ i \in \{1, ..., N\}$. To avoid causing any bias to the algorithm proposed in [34], we assume that all of the offloaded applications decided by their algorithm are offloaded to the cloud server, which is similar to the scenario that they considered.

In our experiments, the arrival of applications on a mobile device follows a Poisson distribution. The execution times on mobile devices and servers, the input data size and the output data size follow negative exponential distributions that are the same as those in [33, 34]. Hence, the execution of applications at a mobile device is treated as an M/M/1 queue [66]; the uplink transmission is also treated as an M/M/1 queue. According to Burke's theorem [67], the output processes of M/M/1 and M/M/∞ also follow a Poisson distribution. Thus, the execution of applications at the edge server and the downlink transmission are also treated as

M/M/1 queues. Then, we have Eqs. (4.71)–(4.74).

$$h_i(\alpha_i) = \frac{(1-\alpha_i)\theta_i}{\mu - (1-\alpha_i)\theta_i}, \quad i \in \{1, ..., N\}, \tag{4.71}$$

$$h_U(\lambda, u) = \frac{\lambda}{\mu_u - \lambda} = \frac{\lambda}{\frac{u}{D_u} - \lambda}, \tag{4.72}$$

$$h_E(\lambda, \alpha_E) = \frac{(1-\alpha_E)\lambda}{\mu_E - (1-\alpha_E)\lambda}, \tag{4.73}$$

$$h_D(\lambda, d) = \frac{\lambda}{\mu_d - \lambda} = \frac{\lambda}{\frac{d}{D_d} - \lambda}. \tag{4.74}$$

We sort the mobile devices by their application arrival rates in descending order, i.e., $\theta_1 \geq \theta_2 \geq ... \geq \theta_N$. Then, u^*, d^*, ${\alpha_E}^*$, ${\alpha_i}^*$, $i \in \{1, ..., N\}$ and φ_j, $j \in \{1, ..., 2N\}$ is calculated as follows.

$$u^* = \frac{(D_u\sqrt{D_d} - D_d\sqrt{D_u})\lambda + \sqrt{D_u}S}{\sqrt{D_u} + \sqrt{D_d}}, \tag{4.75}$$

$$d^* = \frac{(D_d\sqrt{D_u} - D_u\sqrt{D_d})\lambda + \sqrt{D_d}S}{\sqrt{D_u} + \sqrt{D_d}}, \tag{4.76}$$

$${\alpha_E}^* = \begin{cases} \begin{cases} 0, \\ \quad \text{if } \lambda \leq \mu_E - \sqrt{\frac{\mu_E\mu_C}{1+2H\mu_C}}, \\ 1 - \frac{1}{\lambda}\left(\mu_E - \sqrt{\frac{\mu_E\mu_C}{1+2H\mu_C}}\right), \\ \quad \text{otherwise}, \end{cases} & \text{if } \frac{1}{\mu_E} \leq \frac{1}{\mu_C} + 2H, \\ 1, & \text{otherwise}, \end{cases} \tag{4.77}$$

$${\alpha_i}^* = \begin{cases} 0, & \text{if } \lambda \leq \sum_{j=1}^{i}(\theta_j - \theta_i), \\ \frac{1}{k\theta_i}\left(\lambda - \sum_{j=1}^{k}(\theta_j - \theta_i)\right), & \\ & \text{if } \sum_{j=1}^{k}(\theta_j - \theta_k) < \lambda \leq \sum_{j=1}^{k+1}(\theta_j - \theta_{k+1}), \\ & \quad k \in \{i, ..., N-1\}, \\ 1, & \text{otherwise}, \end{cases} \tag{4.78}$$

$$\varphi_j = \begin{cases} \sum_{i=1}^{j}(\theta_i - \theta_j), & \text{if } i \leq N, \\ \sum_{i=1}^{N}\theta_i, & \text{otherwise.} \end{cases} \tag{4.79}$$

$f_1{}^*(\lambda)$, $f_2{}^*(\lambda)$ and $f_3{}^*(\lambda)$ can be calculated by $f_1{}^*(\lambda) = h_E(\lambda, \alpha_E{}^*) + h_C(\lambda, \alpha_E{}^*)$, $f_2{}^*(\lambda) = h_E(\lambda, \alpha_E{}^*) + h_C(\lambda, \alpha_E{}^*)$ and $f_3{}^*(\lambda) = \sum_{i=1}^{N} h_i(\alpha_i)$, respectively, and shown as follows.

$$f_1{}^* = \frac{(\sqrt{d_U} + \sqrt{d_D})^2\lambda}{S - (d_U + d_D)\lambda}, \tag{4.80}$$

$$f_2{}^* = \begin{cases} \begin{cases} \frac{\lambda}{\mu_E - \lambda}, \\ \qquad \text{if } \lambda \leq \mu_E - \sqrt{\frac{\mu_E}{\frac{1}{\mu_C} + 2H}}, \\ (\lambda - \mu_E)(\frac{1}{\mu_C} + 2H) - 1 + 2\sqrt{\frac{\mu_E}{\mu_C} + 2\mu_E H}, \\ \qquad \text{otherwise,} \end{cases} & \text{if } \frac{1}{\mu_E} \leq \frac{1}{\mu_C} + 2H, \\ \lambda(\frac{1}{\mu_C} + 2H), & \text{otherwise,} \end{cases} \tag{4.81}$$

$$f_3{}^* = \begin{cases} \frac{\theta_1 - \lambda}{\mu - (\theta_1 - \lambda)} + \sum_{i=2}^{N} \frac{\theta_i}{\mu - \theta_i}, & \text{if } \lambda \leq \theta_1 - \theta_2, \\ \frac{j\sum_{i=1}^{j}\theta_j - j\lambda}{j\mu - \sum_{i=1}^{j}\theta_i - \lambda} + \sum_{i=j+1}^{N} \frac{\theta_i}{\mu - \theta_i}, & \text{if } \sum_{i=1}^{j}(\theta_i - \theta_j) \leq \lambda \leq \sum_{i=1}^{j+1}(\theta_i - \theta_{j+1}), \\ & j \in \{2, ..., N-1\}, \\ \frac{N\sum_{i=1}^{N}\theta_i - N\lambda}{N\mu - \sum_{i=1}^{N}\theta_i - \lambda}, & \text{if } \sum_{i=1}^{N}(\theta_i - \theta_N) \leq \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{4.82}$$

4.3.1 Parameter Settings

The number of mobile devices is set to 100. The service rate of a mobile device and the average size of the application input data are set to 1 and 0.4 MB, respectively, i.e., $\mu = \frac{5}{7}$ and $D_u = 0.4$ MB. Additionally, the average size of the application

output data is set as 0.1 MB, i.e., $D_d = 0.1$ MB. The service rate of the edge server is set to 14 times that of a mobile device, i.e., $\frac{\mu_E}{\mu} = 14$. We assume that the service rate of the cloud server is the same as that of the edge server, i.e., $\mu_E = \mu_C$. The total transmission rate is set as 80 Mbps, i.e., $S = 80$ Mbps. The transmission delay of the high-speed link between the edge server and the cloud server is set to 0.5 s, i.e., $H = 0.5$ s. The average arrival rate of applications on the ith mobile device, $\theta_i, i \in \{1, ..., N\}$, is generated randomly within the range [0, 0.8μ]. The setting of parameters is shown in Table 4.4. The program used in our experiments was developed using Python 3.6 on a computer running Microsoft Windows 10 with an Intel E3 3.0 GHz CPU and 24 GB of memory.

Table 4.4. Parameter settings in experiments.

Setting	Description
100	number of mobile devices
$\frac{5}{7}$	service rate of a mobile device
14	service rate ratio of server vs. mobile device
0.4MB	average sizes of the input data
0.1MB	average sizes of the output data
80Mbps	total transmission rate
0.5s	transmission delay between the edge server and the cloud server
$[0, 0.8 \times \frac{5}{7}]$	average arrival rate of applications on a mobile device

4.3.2 Effect of Total Transmission Rate

Figure 4.10 shows the average application response times for different approaches with various total transmission rates of the shared channels, as obtained through numerical experiments. The total transmission rates are set as 0, 20, 40, 60, 80, 100, 120, 140 and 160 Mbps in each experiment, respectively. We find that our proposed algorithm performs much better than the others for a wide range of total transmission rates. When the total transmission rate is low, few applications are offloaded. As the total transmission rate increases, increasingly more applications are offloaded. When the total transmission rate is low, the IPM-based algorithm yields considerably worse results than the other algorithms because the IPM-based algorithm does not consider the queuing time on the shared communication, resulting in incorrect offloading. Additionally, when the total transmission rate is high, as the transmission rate increases, the performance of the IPM-based algorithm is almost invariant because it fails to consider the cooperation between the edge server and the cloud server. The limited resources of the edge server restrict

the application offloading in the IPM-based algorithm. Because the ERWP algorithm fails to consider the cooperation of the edge server and the cloud server, our algorithm always outperforms the ERWP algorithm. These results demonstrate that our algorithm can choose an opportune offloading strategy for various total transmission rates.

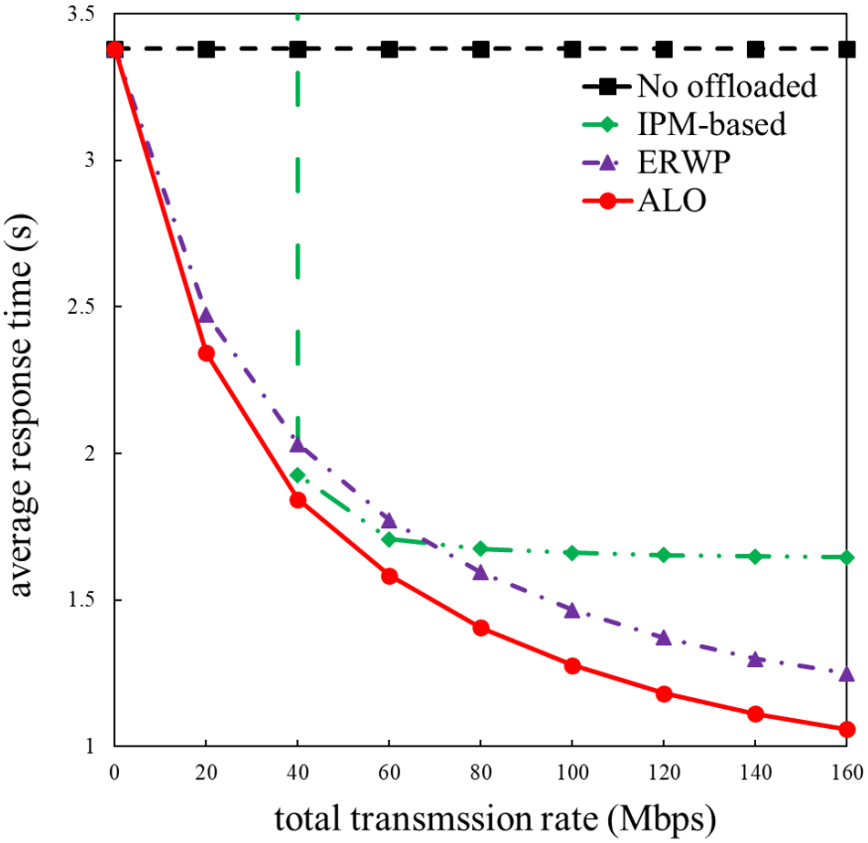


Figure 4.10. Effect of total transmission rate.

In order to further test the performance of our proposed algorithm, we consider the case of that the service rates of mobile devices are different with each other. In this case, the service rate of a mobile device is generated randomly within the range $[\frac{4}{7}, \frac{6}{7}]$ rather than fixed as $\frac{5}{7}$. The average application response times for different approaches with various total transmission rates, as obtained through numerical experiments, are shown in Figure 4.11. We can find that our algorithm still outperforms previous algorithms no matter whether the service rates of mobile devices are same or not.

Additionally, in order to test the robustness of our proposed algorithm, we consider the case of that transmission rates are randomly fluctuated within the ranges

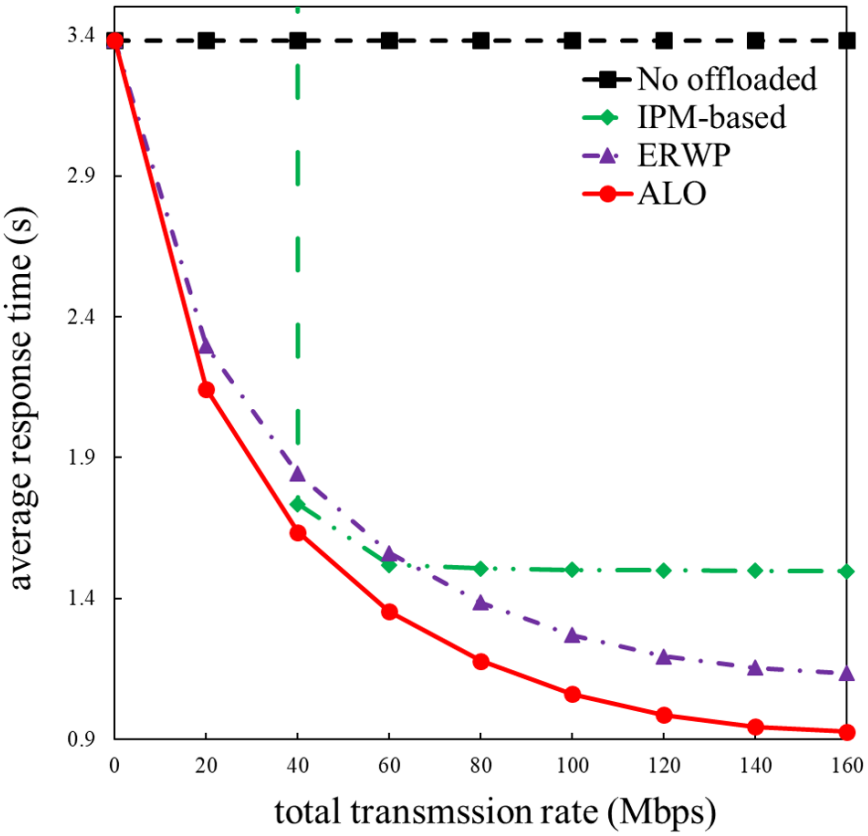


Figure 4.11. Effect of total transmission rate (service rates of mobile devices are different).

of [20, 60], [40, 120], [60, 180] and [80, 240] Mbps, respectively. The offloading strategies are obtained with fixed total transmission rate as 40, 80, 120 and 160 Mbps, respectively, but tested in fluctuating transmission rate scenarios. We simulate the execution of applications that arrive over a period of 12 hours, where each simulation is run 100 times. Figure 4.12 shows the average response times of applications when the transmission rate fluctuation time ratio changes from 0 to 100%. The half-widths of the confidence intervals at the 99% confidence level are all less than 0.1% of the sample means and therefore are not shown in the figure. We find that the response time is nearly invariant for all algorithms in each scenarios, proving the high robustness of these algorithms.

Moreover, we also consider an other case of transmission rate fluctuation. In a period of 1000 seconds, the total transmission rate increases by 10% in a time slot of 100 seconds, but decreases by 10% in another time slot of 100 seconds. The offloading strategies are obtained with fixed total transmission rate but tested in

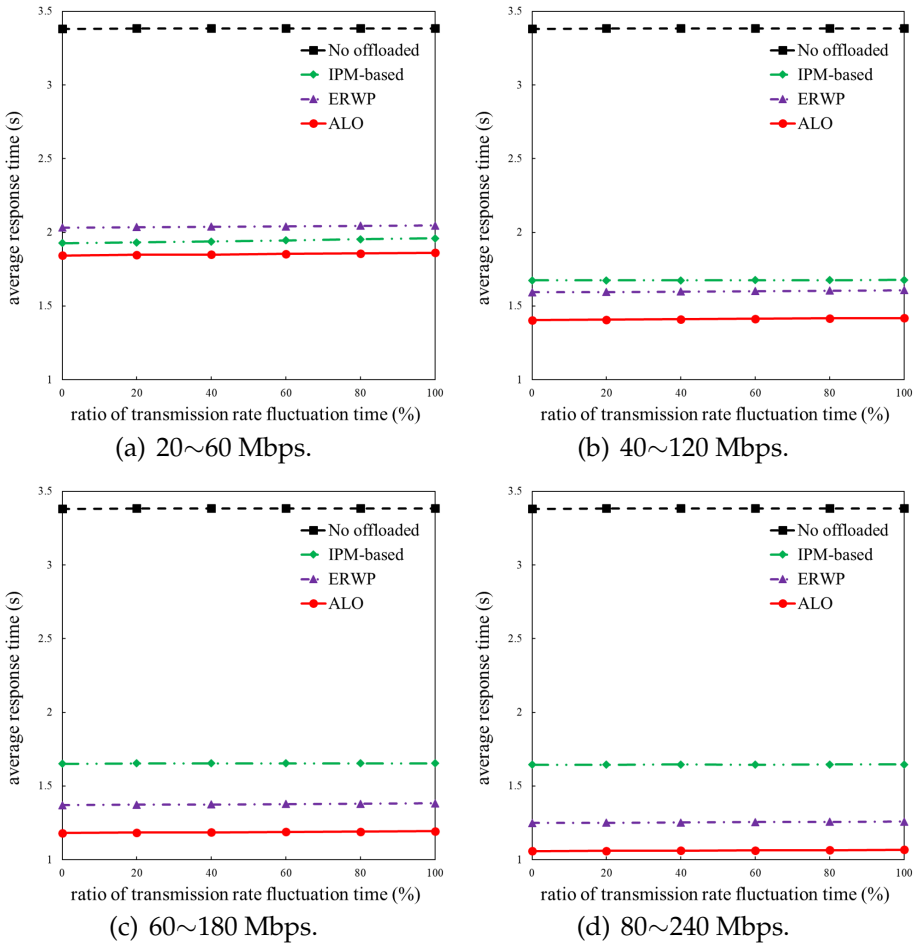
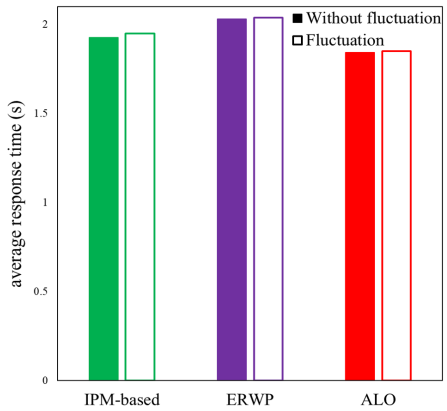


Figure 4.12. Randomly fluctuating total transmission rate.

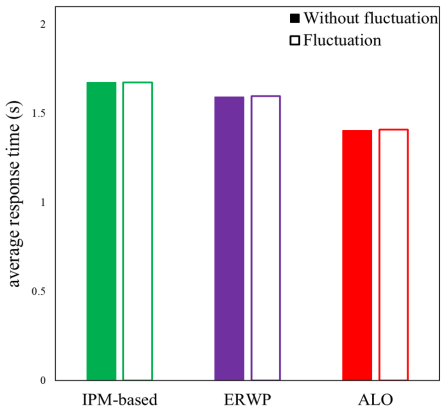
fluctuating transmission rate scenarios. We simulate the execution of applications that arrive over a period of 12 hours, where each simulation is run 100 times. Figure 4.13 shows the average response times of applications in which the average transmission rates are 40, 80, 120 and 160 Mbps, respectively. The half-widths of the confidence intervals at the 99% confidence level are all less than 0.1%. The results show that fluctuation ratios of average response time are lower than 1%.

4.3.3 Effect of Ratio of the Computing Power of Server vs. Mobile device

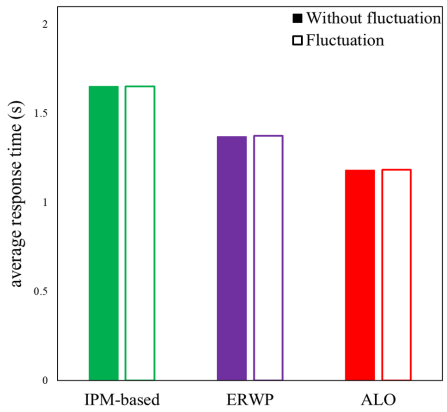
Figure 4.14 shows the average application completion times achieved with the different approaches for various values of the ratio of the computing power of the



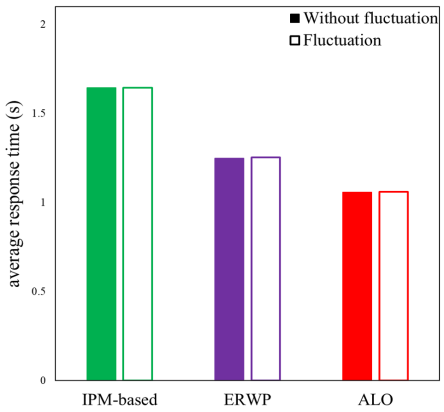
(a) Average transmission rate is 40 Mbps.



(b) Average transmission rate is 80 Mbps.



(c) Average transmission rate is 120 Mbps.



(d) Average transmission rate is 160 Mbps.

Figure 4.13. Continuously fluctuating total transmission rate.

server to that of the mobile device, as obtained through numerical experiments. The values of ratio of the computing power of a server to that of a mobile device are set as 6, 8, 10, 12, 14, 16, 18 and 20 in each experiment, respectively. From Figure 4.14, we find that our algorithm can always achieve high performance since the cooperation between servers is considered. As the computing power ratio between the server and mobile device increases, the average response time of the IPM-based algorithm decreases because this algorithm only utilized the computation resources on the edge server, which causes a large queueing time on the edge server when the processing rate of the server is low. The performance of the ERWP algorithm is almost invariant since this algorithm does not consider the cooperation of the edge server and the cloud server. These results demonstrate that

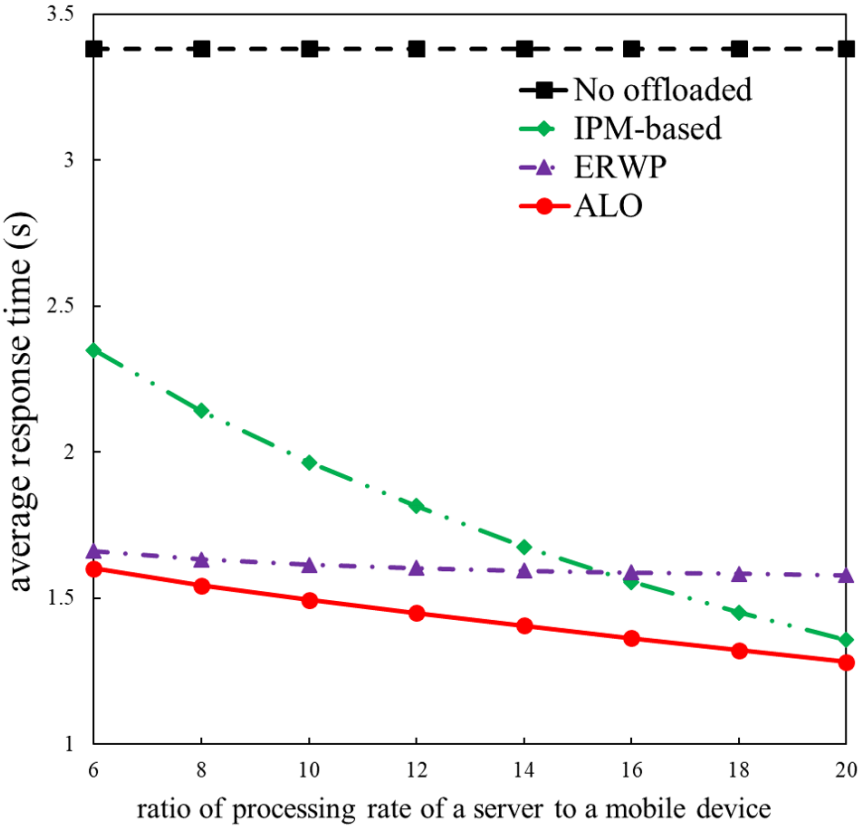


Figure 4.14. Effect of computing power ratio.

our algorithm ensures the utilization of server resources while ensuring that the transmission delay remains low.

4.3.4 Effect of Number of Mobile Devices

Figure 4.15 shows the average application completion times of algorithms with varying numbers of mobile devices, as obtained through numerical experiments. The numbers of users are set as 1, 50, 100, 150 and 200 in each experiment, respectively. From Figure 4.15, we can find that our algorithm outperforms the other algorithms. When there are few mobile devices, all of the algorithms offload all applications from the mobile devices. However, applications are offloaded to the remote cloud server in the ERWP algorithm, which cases extra transmission delay. Thus, our algorithm and the IPM-based algorithm outperform the ERWP algorithm. As the number of mobile devices increases, the ERWP algorithm shows better performance than the IPM-based algorithm but still performs worse than

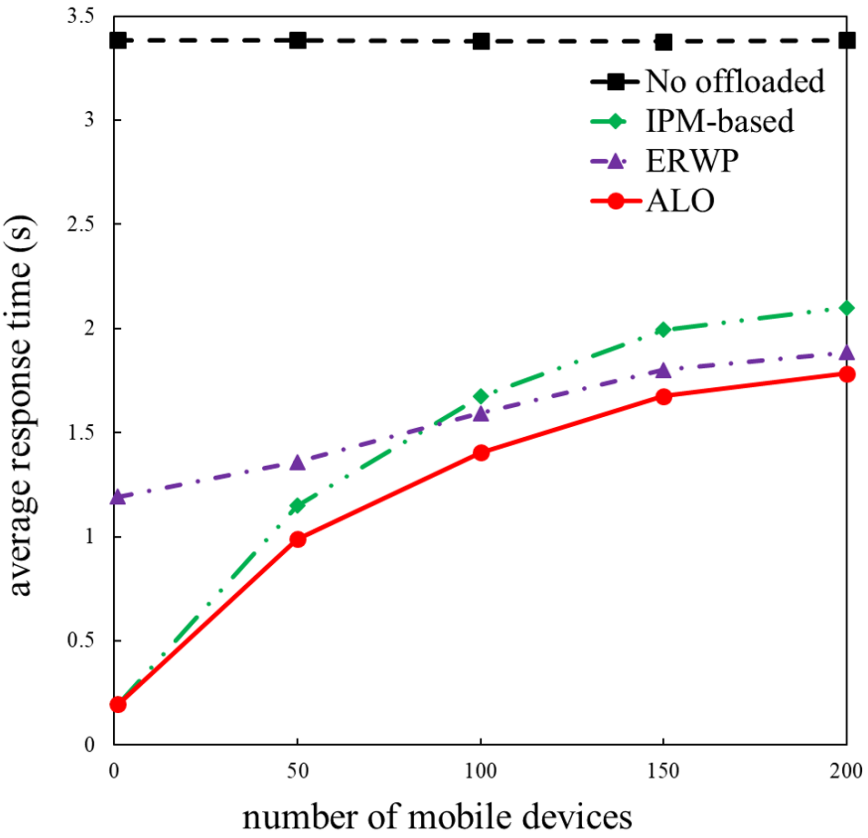


Figure 4.15. Effect of the number of users.

our algorithm. The cooperation between servers is considered in our algorithm so that the utilization of server resources in our algorithm is higher than that in the other algorithms. These results prove that our algorithm shows better scalability than the other algorithms.

4.4 Conclusion

In this study, we focus on the problem of how to offload computationally intensive applications and how to determine the transmission rate of communication channels in order to minimize the average application response time. We consider a three-tier cloud computing system in which applications are offloaded from mobile devices to the edge servers and further offloaded to the cloud servers if necessary. We also consider the limited total transmission rate of the shared communication channels. The problems of computation offloading and transmission

rate determination have been jointly studied and formulated as an optimization problem. The joint problem has been analysed and transformed into a piecewise convex optimization problem. An efficient algorithm has been proposed to find the optimal solution to our problem. The proposed algorithm has been tested by experiments, and the results show that our proposed algorithm outperforms previous algorithms. Additionally, the results also show that our algorithm has high robustness.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we firstly study the task level offloading problem which controls both of the data transmission and the task execution. Then, we study the application level offloading problem from the viewpoint of system analysis. In conclusion, the computation offloading for mobile cloud computing in term of minimizing the average response time of applications has been completely studied in this thesis.

In chapter 3, we have studied the task level offloading problem of how to offload tasks of applications, which consist of a set of parallel and serial tasks, from multiple mobile devices to an edge server via a shared communication channel in an online scenario while considering the resource limitations of both of the shared channel and the edge server. A three-phase offloading method is proposed in this study to solve the problem. We firstly focus on the offloading problem where only the resource limitation of the shared channel is considered. Two effective offloading approaches are proposed and switched depending on the degree of system congestion in order to obtain the initial solution. In second phase, the resource limitation of the edge server is considered and a resource assignment algorithm is proposed to assign the unoccupied computation resource for tasks which are decided to be offloaded in initial solution and adjust the initial solution. At last, the data transmission time is rescheduled if necessary. We have tested the proposed method via simulation experiments with various system parameters and find that our method is significantly outperforms previous methods. The main reason of this finding is that the collisions on the shared channel are scheduled and the characteristic of parallel tasks is utilized in our offloading strategy.

In chapter 4, we have studied the application level offloading problem of how to integrally offload applications from mobile devices. A three-tier computation offloading system with multiple mobile devices, edge servers and cloud servers is

considered in our study. In this system, an application can be integrally offloaded from a mobile device to an edge server, and further offloaded to a cloud server if necessary. The resource limitations of the mobile devices, edge servers and shared channels between mobile devices and edge servers are considered in our study. We focus on the joint computation offloading and transmission rate determination problem and propose an offline offloading strategy to minimize the average response time of applications. We have formulated the joint problem as a non-convex optimization problem based on the characteristic of the computation offloading system and further transform the formulated problem into a single-variable piecewise convex optimization problem based on our analysis. An efficient algorithm is proposed to find the optimal offloading ratios and the optimal transmission rates. According to experiments, we can find that our proposed algorithm outperforms previous algorithms and the performance of our algorithm is highly robust.

5.2 Future Work

In this thesis, we have studied the computation offloading for mobile cloud computing in term of minimizing time consumption. In this section, we will present some worthy research topics in mobile cloud computing which can be studied in the future.

- Computation offloading in term of minimizing energy consumption.

  We have studied the computation offloading problem to minimize time consumption in this thesis. However, it is also important to reduce the energy consumption for mobile devices. Because of the limited battery capacity on mobile devices, one of the most crucial design objectives of the mobile device is extending its battery lifetime [68]. Some related works such as [69–73] have focused on the energy consumption minimization. Being different with time consumption minimization, only the energy consumption of mobile devices, such as the energy consumption for computation processing and the data transferring/receiving, are taken into consideration in this topic. Therefore, the offloading strategies which aim at minimizing energy consumption prefer to offload more computations to the server in order to reduce the energy consumption of the mobile devices. However, it may postpone response time of applications. Thus, it is important to study how to minimize the energy consumption under a response time requirement.

- Data sharing in mobile cloud computing.

  The problem of how to use the computation resources of servers has been studied in this thesis. However, servers can provide not only the computa-

tion resources but also the storage resources. It is also important to study how to use the storage resources of servers, i.e., data sharing problem in mobile cloud computing. There are some related works such as [74–80] have studied the data sharing in mobile cloud computing. Being different with computation offloading, which is an unidirectional process that the computations are offloaded from a mobile device to a server, data sharing is a bidirectional process including both of data collecting from a mobile device to a server and data pushing from a server to a mobile device. In data collection, it is necessary to study how to enhance the security and reduce the energy consumption of data transmission. In data pushing, the major problem is how to allocate the data at servers in order to ensure that the data can be transmitted to mobile devices efficiently.

Acknowledgements

First and foremost, I would like to express my the deepest respect and the most sincere appreciation to my adviser, Prof. Yongbing Zhang, for his patience, encouragement and careful guidance. Every time I have questions, he always give me careful guidance and detailed advice. When I am confused, he always points my way forward. Without his enlightening instruction, I could not have completed my thesis. His boundless enthusiasm, careful attention to detail and infinite patience enlighten me not only in this thesis but also in my future study. It is my privilege to have the opportunity to study under his guidance.

I also would like to thank Prof. Phung-Duc Tuan and Prof. Masahiro Hachimori for their useful suggestions and opinions. According to their advices, I can finish my study smoothly.

In addition, I would like to thank the rest of my thesis committee: Prof. Hiroyasu Ando and Prof. Shigetomo Kimura, for their insightful comments and instructive suggestions.

Last but not least, I would like to extend my special thanks to my parents and my family, who support me to move on and give me courage in my difficult times.

Bibliography

[1] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, No. 4, 2010; pp. 51-56.

[2] K. Akherfi, M. Gerndt and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied computing and informatics*, Vol. 14 No. 1, 2018; pp. 1-16.

[3] S. Patidar, D. Rane and P. Jain, "A survey paper on cloud computing," *2012 Second International Conference on Advanced Computing & Communication Technologies*, 2012; pp. 394-398.

[4] X. Fan, J. Cao, and H. Mao, "A survey of mobile cloud computing," *ZTE Communications*, 2011.

[5] F. Liu, P. Shu, H. Jin, et al. "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications*, Vol. 20, No. 3, 2013; pp. 14-22.

[6] F. Wortmann and K. Flüchter, "Internet of things," *Business & Information Systems Engineering*, Vol. 57, No. 3, 2015; pp. 221-224.

[7] B. Liang, V. W. S. Wong, R. Schober, et al. "Mobile edge computing," *Key Technologies for 5G Wireless Systems*, Vol. 16, No. 3, 2017; pp. 1397-1411.

[8] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, Vol. 68, No. 1, 2019; pp. 856-868.

[9] V. Tripathi, "Adaptive Computation Offloading in Mobile Cloud Computing," *CLOSER*, 2017.

[10] A. Amies, "Developing and hosting applications on the cloud," *IBM Press*, 2012.

[11] W. Y. Chang, H. Abu-Amara and J. F. Sanford, "Transforming enterprise cloud services," *Springer Science & Business Media*, 2010.

[12] A. Fox, D. A. Patterson and S. Joseph, "Engineering software as a service: an agile approach using cloud computing," *Strawberry Canyon LLC*, 2013.

[13] Z. Li, C. Wang and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," *ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 2001; pp. 238-246.

[14] M. V. Barbera, S. Kosta, A. Mei, et al. "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," *IEEE International Conference on Computer Communications*, 2013; pp. 1285-1293.

[15] I. Trummer and C. Koch, "Solving the join ordering problem via mixed integer linear programming," *ACM International Conference on Management of Data*, 2017; pp. 1025-1040.

[16] O. Gurobi, "Gurobi optimizer reference manual," *http://www.gurobi.com*, 2015.

[17] B. Zhao, Z. Xu, C. Chi, et al. "Mirroring smartphones for good: A feasibility study," *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 2010; pp. 26-38.

[18] R. Kemp, N. Palmer, T. Kielmann, et al. "Cuckoo: a computation offloading framework for smartphones," *International Conference on Mobile Computing, Applications, and Services*, 2010; pp. 59-79.

[19] E. Cuervo, A. Balasubramanian, D. Cho, et al. "MAUI: making smartphones last longer with code offload," *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 2010, pp. 49-62.

[20] M. Satyanarayanan, V. Bahl, R. Caceres, et al. "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, 2009.

[21] M. R. Rahimi, "Exploiting an elastic 2-tiered cloud architecture for rich mobile applications," *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2012; pp. 1-2.

[22] R. M. Rahimi, N. Venkatasubramanian, S. Mehrotra, et al. "MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," *IEEE/ACM 5th International Conference on Utility and Cloud Computing*, 2012; pp. 83-90.

[23] M. R. Rahimi, N. Venkatasubramanian and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," *IEEE 6th International Conference on Cloud Computing*, 2013; pp. 75-82.

[24] T. X. Tran, A. Hajisami, P. Pandey, et al. "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, Vol. 55, No. 4, 2017; pp. 54-61.

[25] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 26, No. 4, 2014; pp. 974-983.

[26] X. Chen, L. Jiao, W. Li, et al. "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, Vol. 24, No, 5, 2015; pp. 2795-2808.

[27] E. Meskar, T. D. Todd, D. Zhao, et al. "Energy efficient offloading for competing users on a shared communication channel," *IEEE International Conference on Communications*, 2015; pp. 3192-3197.

[28] E. Meskar, T. D. Todd, D. Zhao, et al. "Energy aware offloading for competing users on a shared communication channel," *IEEE Transactions on Mobile Computing*, Vol. 16, No. 1, 2016; pp. 87-96.

[29] J. Zheng, Y. Cai, Y. Wu, et al. "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, Vol. 18, No. 4, 2018; pp. 771-786.

[30] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," *IEEE Conference on Computer Communications*, 2017; pp. 1-9.

[31] C. Yi, J. Cai and Z. Su, "A Multi-User Mobile Computation Offloading and Transmission Scheduling Mechanism for Delay-Sensitive Applications," *IEEE Transactions on Mobile Computing*, 2019.

[32] W. Fan, Y. Liu, B. Tang, et al. "Computation offloading based on cooperations of mobile edge computing-enabled base stations," *IEEE Access*, Vol. 6, 2017; pp. 22622-22633.

[33] L. Liu, Z. Chang, X. Guo, et al. "Multi-objective optimization for computation offloading in mobile-edge computing," *IEEE Symposium on Computers and Communications*, 2017; pp. 832-837.

[34] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, Vol. 17, No. 2, 2017; pp. 461-474.

[35] X. Ma, S. Wang, S. Zhang, et al. "Cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing," *IEEE Transactions on Cloud Computing*, 2019.

[36] M. T. Thai, Y. D. Lin, Y. C. Lai, et al. "Workload and Capacity Optimization for Cloud-Edge Computing Systems with Vertical and Horizontal Offloading", *IEEE Transactions on Network and Service Management*, 2019.

[37] B. G. Chun, S. Ihm, P. Maniatis, et al. "Clonecloud: elastic execution between mobile device and cloud," *ACM Proceedings of the 6th Conference on Computer systems*, 2011; pp. 301-314.

[38] M. R. Ra, A. Sheth, L. Mummert, et al. "Odessa: enabling interactive perception applications on mobile devices," *ACM Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, 2011; pp. 43-56.

[39] M. Jia, J. Cao and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," *IEEE Conference on Computer Communications Workshops*, 2014; pp. 352-357.

[40] H. Wu, W. Knottenbelt, K. Wolter, et al. "An optimal offloading partitioning algorithm in mobile cloud computing," *International Conference on Quantitative Evaluation of Systems*, 2016; pp. 311-328.

[41] M. Goudarzi, M. Zamani and A. T. Haghighat, "A fast hybrid multi-site computation offloading for mobile cloud computing," *Journal of Network and Computer Applications*, Vol. 80, 2017; pp. 219-231.

[42] Y. Tao, Y. Zhang and Y. Ji, "Efficient computation offloading strategies for mobile cloud computing," *IEEE International Conference on Advanced Information Networking and Applications*, 2015; pp. 626-633.

[43] R. Kumari, S. Kaushal and N. Chilamkurti, "Energy conscious multi-site computation offloading for mobile cloud computing," *Soft Computing*, Vol. 22, No. 20, 2018; pp. 6751-6764.

[44] G. Zhang, Y. Chen, Z. Shen, et al. "Distributed energy management for multi-user mobile-edge computing systems with energy harvesting devices and QoS constraints." *IEEE Internet of Things Journal*, 2018.

[45] Y. Mao, J. Zhang and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," *IEEE Wireless Communications and Networking Conference*, 2017; pp. 1-6.

[46] H. Xing, L. Liu, J. Xu, et al. "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," *IEEE International Conference on Communications*, 2018; pp. 1-6.

[47] H. Wu, Y. Sun and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, 2018.

[48] L. Yang, J. Cao, H. Cheng, et al. "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, Vol. 64, No. 8, 2014; pp. 2253-2266.

[49] L. Yang, B. Liu, J. Cao, et al. "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," *IEEE Transactions on Services Computing*, 2019.

[50] K. Guo, M. Yang, Y. Zhang Y, et al. "An Efficient Dynamic Offloading Approach based on Optimization Technique for Mobile Edge Computing," *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2018; pp. 29-36.

[51] K. Guo, M. Yang and Y. Zhang, "Computation offloading over a shared communication channel for mobile cloud computing", *IEEE Wireless Communications and Networking Conference*, 2018; pp. 1-6.

[52] K. Guo, M. Yang, Y. Zhang, et al. "Efficient resource assignment in mobile edge computing: A dynamic congestion-aware offloading approach", *Journal of Network and Computer Applications*, Vol. 134, 2019; pp. 134: 40-51.

[53] P. Dempsey, "The Teardown: Samsung Galaxy S8 Plus smartphone," *Engineering & Technology*, Vol. 12, No. 6, 2017; pp. 78-79.

[54] L. Yang, W. Liu, N. Guan, et al. "Dark silicon-aware hardware-software collaborated design for heterogeneous many-core systems," *22nd Asia and South Pacific Design Automation Conference*, 2017; pp. 494-499.

[55] S. Yang, X. Bei, Y. Zhang, et al. "Application offloading based on R-OSGi in mobile cloud computing," *IEEE 4th International Conference on Mobile Cloud Computing, Services, and Engineering*, 2016; pp. 46-52.

[56] Y. Zhang, H. Liu, L. Jiao, et al. "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing," *IEEE 1st International Conference on Cloud Networking*, 2012; pp. 80-86.

[57] A. B. Craig, "Understanding augmented reality: Concepts and applications," *Newnes*, 2013.

[58] E. Blem, J. Menon and K. Sankaralingam, "A detailed analysis of contemporary arm and x86 architectures," *UW-Madison Technical Report*, 2013.

[59] K. Guo, M. Yang, Y. Zhang, et al. "Joint Computation Offloading and Bandwidth Assignment in Cloud-Assisted Edge Computing", *IEEE Transactions on Cloud Computing*, 2019.

[60] J. G. Shanthikumar, S. Ding and M. T. Zhang, "Queueing theory for semiconductor manufacturing systems: a survey and open problems," *IEEE Transactions on Automation Science and Engineering*, Vol. 4, No. 4, 2007; pp. 513-522.

[61] Chain rule, https://en.wikipedia.org/wiki/Chain_rule

[62] Lagrange multiplier, https://en.wikipedia.org/wiki/Lagrange_multiplier

[63] Karush–Kuhn–Tucker conditions, https://en.wikipedia.org/wiki/Karush-Kuhn-Tucker_conditions

[64] Binary search algorithm, https://en.wikipedia.org/wiki/Binary_search_algorithm

[65] Newton's method, https://en.wikipedia.org/wiki/Newton%27s_method

[66] J. Abate and W. Whitt, "Transient behavior of the M/M/1 queue: Starting at the origin," *Queueing systems*, Vol. 2, No. 1, 1987; pp. 41-65.

[67] J. Walrand, "A probabilistic look at networks of quasi-reversible queues," *IEEE transactions on information theory*, Vol. 29, No. 6, 1983; pp. 825-831.

[68] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, Vol. 6, 2018; pp. 3962-3976.

[69] G. Zhang, W. Zhang, Y. Cao, et al. "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 10, 2018; pp. 4642-4655.

[70] Y. Zhang, J. He and S. Guo, "Energy-Efficient Dynamic Task Offloading for Energy Harvesting Mobile Cloud Computing," *IEEE International Conference on Networking, Architecture and Storage*, 2018; pp. 1-4.

[71] Y. Kim, H. W. Lee and S. Chong, "Mobile computation offloading for application throughput fairness and energy efficiency," *IEEE Transactions on Wireless Communications*, Vol. 18, No. 1, 2018; pp. 3-19.

[72] C. You, Y. Zeng, R. Zhang, et al. "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," *IEEE Transactions on Wireless Communications*, Vol. 17, No. 11, 2018; pp. 7590-7605.

[73] J. Zhang, H. Guo and J. Liu, "Energy-Aware Task Offloading for Ultra-Dense Edge Computing," *IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, 2018; pp. 720-727.

[74] J. Zhang, Z. Zhang and H. Guo. "Towards secure data distribution systems in mobile cloud computing," *IEEE Transactions on Mobile Computing*, Vol. 16, No. 11, 2017; pp. 3222-3235.

[75] R. Li, C. Shen, H. He, et al. "A lightweight secure data sharing scheme for mobile cloud computing," *IEEE Transactions on Cloud Computing*, Vol. 6, No. 2, 2017; pp. 344-357.

[76] X. Li, S. Liu, F. Wu, et al. "Privacy preserving data aggregation scheme for mobile edge computing assisted IoT applications," *IEEE Internet of Things Journal*, 2018.

[77] Y. Xie, H. Wen, B. Wu, et al. "A modified hierarchical attribute-based encryption access control method for mobile cloud computing," *IEEE Transactions on Cloud Computing*, Vol. 7, No, 2, 2019; pp. 383-391.

[78] R. K. Lomotey, J. A. Nilson, K. Mulder, et al. "Mobile medical data synchronization on cloud-powered middleware platform," *IEEE Transactions on Services Computing*, Vol. 9, No. 5, 2016; pp. 757-770.

[79] A. Awad, A. Matthews, Y. Qiao, et al. "Chaotic searchable encryption for mobile cloud storage," *IEEE Transactions on Cloud Computing*, Vol. 6, No. 2, 2015; pp. 440-452.

[80] A. Capponi, C. Fiandrino, D. Kliazovich, et al. "A cost-effective distributed framework for data collection in cloud-based mobile crowd sensing architectures," *IEEE Transactions on Sustainable Computing*, Vol. 2, No. 1, 2017; pp. 3-16.