An Efficient Dynamic Offloading Approach based on Optimization Technique for Mobile Edge Computing

Kai Guo*, Mingcong Yang*, Yongbing Zhang†, and Yusheng Ji‡

*†Graduate School of Systems and Information Engineering, University of Tsukuba, Japan
‡Information Systems Architecture Science Research Division, National Institute of Informatics, Japan
Email: *{s1730141, s1730144}@s.tsukuba.ac.jp, †ybzhang@sk.tsukuba.ac.jp, ‡kei@nii.ac.jp

***Abstract*—Mobile Cloud Computing (MCC) is a new paradigm to provide computation capabilities at the edge networks near mobile devices to speed up the executions of mobile applications and reduce the power consumptions at mobile devices. In this paper, we focus on the problem of how to dynamically offload the computation intensive tasks of newly executed mobile applications from resource-scarce mobile devices to the servers located at the edge networks in order to minimize the average application completion time. We consider a system model in which there are a set of mobile devices connected to an edge server by a shared communication network. Furthermore, we take into account of possible transmission collisions at the shared network when more than one mobile device attempts to transmit data simultaneously. We first formulate a static offloading problem as an MIP problem that can be solved using an appropriate optimization tool such as Gurobi optimizer. Then, we extend the static problem to solve the dynamic offloading problem considered in this paper. Using simulation experiments, we show that our proposed approach outperforms significantly previous approaches for a wide range of system parameters.**

I. Introduction

The edge computing paradigm has been attracting interest from researchers in speed-up for both data- and time-intensive mobile applications and in energy saving of mobile devices such as sensors and smartphones. The servers located on the edge networks, called the *edge servers*, extend the computation and storage capabilities of mobile devices and furthermore reduce the power consumptions at mobile devices [1], [2]. Computation intensive tasks are migrated (offloaded) from mobile devices to the edge servers for remote processing [3], [4], [5], [6], [7]. Since the offloading performance depends heavily on not only the characteristics of both the applications and the mobile devices but also the communication networks connecting the mobile devices to the servers, it is crucially important how to offload mobile tasks to the servers. A number of mobile devices are usually connected to an edge server via a shared radio access network such a Wireless LAN and therefore the data transmission from a mobile device to the server may collide with other transmissions over the shared network. The possible collisions over the shared communication network should be taken into account in offloading decisions.

There are some researches focusing on computation offloading [4], [7], [8], [9]. The authors [4], [7] only considered the task offloading for simple applications each of which consists of a single task chain. The optimal offloading strategy can be found for such a single chain application but it cannot be applied to a more complex application such as an application with parallel tasks or with even parallel and sequential mixed tasks. The authors [9], [8] considered the offloading problem for applications with parallel tasks and tried to find the offloading decisions using heuristic method. However, they focused on how to minimize the average execution time of each task in an application instead of the application completion time. In this paper, we aim at minimizing the average completion time for mobile applications with both sequential and parallel tasks.

When the task execution times and the data sizes sent between tasks in each application and furthermore the time instant the application is executed are known in advance, the offloading problem is *static* in nature and can be solved using an Integer Programming (IP) approach. A software package such as the Gurobi Optimizer [10] or the IBM ILOG CPLEX [11] can be used to solve a static offloading problem if the problem size, i.e., the number of tasks in an application and the number of applications in the system, is limited. In this paper, we consider a dynamic offloading problem in the sense that the task execution time and the data transmission size of an application are known in advance but when the application is executed is unknown, i.e., an application can be executed at any time. We formulate the dynamic offloading problem as a Mixed Integer Programming (MIP) problem [12] and attempt to minimize the average completion time of the applications.

The contributions of this paper can be summarized as follows. 1) We formulate a static offloading problem as a MIP problem so that we can use an appropriate software package to obtain its solution. 2) We propose a dynamic offloading approach that offloads a newly executed application on condition that the offloading decisions of all the applications executed previously than the new one are given. The tasks of previous applications that are under either execution or transmission will not be interrupted. By simulation, we see that this approach works quite well when the network load is not high. However, when the network is congested, the computation time for offloading decisions may become extremely long. 3) In order to reduce the computation time, we propose another dynamic offloading approach in which the offloading decisions of the previously executed application are not changed. According to simulation experiments, we see that this approach outperforms a previous approach significantly.

The remainder of this paper is organized as follows. In section 2, we review the related works on computation offloading. In section 3, we describe our system model of application offloading problem. In section 4, we formulate the static offloading problem as a MIP problem. In section 5, we formulate the dynamic offloading problem as a MIP problem and give a dynamic offloading approach for this problem. The performance evaluation of our proposed approach compared with previous algorithms is shown in section 6. Finally, we conclude this paper in section 7.

II. Related Work

There is a promising way to provide powerful computing resources to mobile devices for reducing the application completion time that deploying small-scale servers at the edge of the Internet as shown in Figure 1. Computation-intensive and time-sensitive applications such as mobile augmented reality [9] and face recognition [13] ran at mobile devices can offload their computation to the server on the network to accelerate the application execution and reduce the energy consumption.

Early researches [3], [5] showed the advantages of computation offloading that leads to not only the execution time reduction for mobile applications but also the energy saving at mobile devices. Computation offloading approaches from an application viewpoint are proposed by the authors [14], [15], [16] in which an application is considered as an inseparable job and can be offloaded to server integrally. They formulated the problem of how to reduce the energy consumption as a competition game and each application acted as a competitor who just considered of its own gain. They found a Nash equilibrium for mobile users to offload their applications of not for getting the largest gain. However, there is a truth that application is usually comprised of a number of dependent tasks each of which can be considered as an independent entity for offloading.

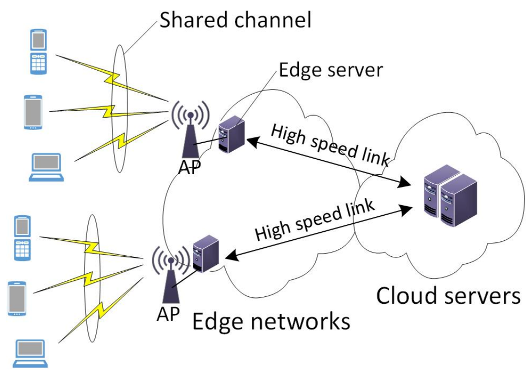

Fig. 1. Edge computing for mobile clouds.

Furthermore, researchers [4], [7], [8], [9] focused on the application offloading for speeding up the execution of the application. Chun *et al.* [4] aimed to pre-calculate an offloading decision before executing each application at a mobile device. However, such a pre-calculated approach is difficult to correspond to an unpredictable dynamic situation. Ra *et al.* [7] proposed a greedy offloading algorithm for mobile applications each of which consists of multiple tasks that are processed serially. Jia *et al.* [8] considered an application with pure parallel or pure serial tasks and proposed a heuristic offloading algorithm that tries to make the load balancing between the mobile device and the cloud server. However, in real case, there always be both parallel tasks and serial tasks in the same application. Wu *et al.* [9] considered with both parallel tasks and serial tasks in the same application and tried to find a better offloading strategy to reduce the application completion time by a heuristic algorithm. Yang *et al.* [17] proposed a heuristic offloading algorithm that aims to minimize the average completion time of applications each of which consists of a task chain. However, they did not consider the collision of the data transmissions and the heuristic algorithm can not find the optimal solution.

Until now, there is no work in the literature taking into account of the parallel property of tasks in general application and the collision between data transmissions over the shared communication channel. Furthermore, all of the researchers tried to solve the problem by heuristic algorithm, which can not get the optimal solution. In this paper, we aim to propose an optimal offloading approach in order to minimize the average completion time of all of applications with the considering of parallel property of tasks and the collision between data transmissions. We consider the application model in which the tasks of an application can be expressed by a general task graph and formulate the offloading problem as a MIP problem. At last, we find the optimal solution from this MIP problem using an optimization solver.

III. System Model

A. *Mobile Cloud Computing System Model*

In this paper, we consider a mobile cloud computing system model as shown in Figure 1. There are a set of mobile devices, denoted by $\mathcal{N} = \{1, 2, \cdots, N\}$, connected to an *edge server*. We assume that each server is deployed at the edge network near the mobile devices and the computing power of an edge server is much stronger than any mobile device. A number of mobile devices are connected to an access point (AP) via a shared radio communication channel, e.g., a wireless network. The AP is connected to the edge server via a high speed link and the transmission delay between AP and edge server can be neglected. We assume that both the edge server and the mobile devices have enough CPU resources so that each task can be executed independently by a dedicated CPU.

B. *Application Model*

We assume that there is only one application executed at each mobile device. For mobile device $i \in \mathcal{N}$, also called *user* i or *application* i, there are a set of tasks constitute application i, which is denoted as $\mathcal{N}^i = \{0, 1, 2, \cdots, M^i\}$. All of these tasks except the start task 0 and the end task M^i can be offloaded to edge server. Without loss of generality, we assume that the execution relationship can be represented by using a directed task flow graph as shown in Figure 2 in which the set of nodes denoted by $\mathcal{N}^i$ indicate the computation tasks of application i. The set of directed arcs denoted by $\mathcal{E}^i$ connecting two nodes indicate data flows from tasks to tasks and a directed arc $e^i_{j,k} \in \mathcal{E}^i$ is used to indicate that after executing task j there are the output data sent to task k of application i. The data size transmitted from task j to k of application i is denoted as $d^i_{j,k}$ and the transmission delay of this data over communication channel is $d^i_{j,k}/s$ where s denotes the transmission rate. Particularly, for two connected tasks j and k of application i which are processed at the same side, either a mobile device or the edge server, we assume that the transmission delay from task j to k can be neglected. We also assume that parallel tasks such as task 1 and task 2 shown in Figure 2 can be processed in parallel at either a mobile device or at the edge server. Furthermore, we assume that a task can be processed only if it receives all the input data from its previous tasks. For example, task 4 can only be processed after it receives the input data from both tasks 1 and 2.

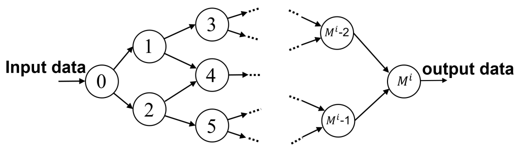

Fig. 2. An application model.

C. Wireless Network Model

The network model considered in this paper is shown in Figure 3 where a number of mobile devices are connected to the edge server via a shared communication channel and may compete with each other for data transmission if they transfer data simultaneously. We assume that the data transmitted based on first-in-first-out (FIFO) principle over the communication channel and an ongoing data transmission cannot be interrupted by any other data transmission. We denote the transmission speed as s.

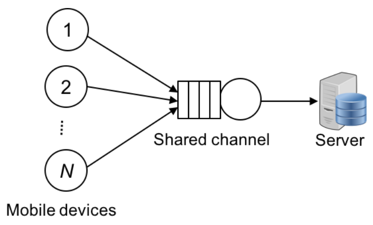

Fig. 3. Network model.

IV. Static Offloading Problem

In this paper, we focus on the problem of how to offload each newly executed application in order to minimize average application completion time. The completion time of an application means the time period from the instant the application is executed at a mobile device to the instant the mobile device receives the output result. In the static offloading problem, we assume that the edge server knows the information of each application such as execution start time of each application, workload of each task, the relationship of tasks and the data transmission size of each data between connected tasks. The server also knows the computing capability of mobile devices so that it can decide which tasks should be offloaded in order to minimize the average application completion time.

According to the execution time of each task and the capacities of both the mobile devices and the server, we can calculate the computation time of task j of application i at the mobile device and the server, denoted by m^i_j and c^i_j, respectively. Since the computing power of the server is greater than any mobile device, we have $m^i_j \geq c^i_j$. The computation start and end times of task j of application i are denoted by τ^i_j and T^i_j, respectively. The start and end times of the data transmission from task j to task k of application i are denoted by $\omega^i_{j,k}$ and $W^i_{j,k}$, respectively. A set of binary integer variables $x^i = \{x^i_0, x_1, \cdots, x^i_{M^i}\}$ are used to show whether the tasks of application i should be offloaded or not. If task j of application i should be offloaded, $x^i_j = 1$ and if it should be executed at mobile device, $x^i_j = 0$. We further assume that tasks 0 and M^i are not eligible for offloading, i.e., $x^i_0 = 0$ and $x^i_{M^i} = 0$. Furthermore, we use an integer variable $y^i_{j,k}$ to show whether two tasks j and k of application i are executed at same side or not. The value of $y^i_{j,k}$ is 0 if task j and task k are executed at the same side, either mobile device i or the server, and 0 otherwise. In order to ensure that there are only one data transmission over the wireless communication channel at the same time, we use a integer variable $z^{i,i'}_{j,k,j',k'}$ to denote whether the data transmission from task j to k of application i is earlier than another data transmission from task j' to k' of application i' or not. The value of $z^{i,i'}_{j,k,j',k'}$ is 1 if $\omega^i_{j,k} \leq \omega^{i'}_{j',k'}$ or 0 otherwise. We let δ^i denote the execution start time of application i. The notation used in this paper is shown in Table I.

In this section, we formulate the static offloading problem as a mixed integer programming (MIP) problem (1) in the sense that the execution time of each application is given in advance. Then, in the next section we extend the static problem to a dynamic offloading problem.

$$\min \quad T = \frac{1}{N}\sum_{i\in\mathcal{N}}\left(T^i_{M^i} - \tau^i_0\right), \tag{1}$$

subject to

$$W^i_{j,k} = \omega^i_{j,k} + y^i_{j,k} d^i_{j,k}/s, \quad j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \tag{2}$$

$$T^i_j = \tau^i_j + x^i_j c^i_j + (1 - x^i_j) m^i_j, \quad j \in \mathcal{N}^i, i \in \mathcal{N} \tag{3}$$

$$\omega^i_{j,k} \geq T^i_j, \ j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \tag{4}$$

$$\tau^i_k \geq W^i_{j,k}, \ j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \tag{5}$$

$$y^i_{j,k} \geq x^i_j - x^i_k, \ j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \tag{6}$$

$$y^i_{j,k} \geq x^i_k - x^i_j, \ j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \tag{7}$$

TABLE I
NOTATION USED IN THIS PAPER.

Symbol	Meaning
$\mathcal{N}$	set of applications
$\mathcal{N}^i$	set of tasks of application i
$\mathcal{E}^i$	set of directed link connecting two tasks of application i
$e^i_{j,k}$	link from task k to task j on the task flow graph of application i
m^i_j	execution time of task j of application i at mobile device
c^i_j	execution time of task j of application i at cloud server
$d^i_{j,k}$	data size transmitted from j to k of application i
τ^i_j	start time of the execution of task j of application i
T^i_j	end time of the execution of task j of application i
$\omega^i_{j,k}$	start time for data transmission from task j to k of application i
$W^i_{j,k}$	end time for data transmission from task j to k of application i
x^i_j	decision variable indicating whether to offload task j of application i to cloud
x^i	decision variables for tasks of application i, i.e., $x^i = \{x^i_j\}, j \in \mathcal{N}^i$
$y^i_{j,k}$	a binary integer variable indicating whether two consecutive tasks j and k of application i are executed at different sides, either at mobile device or at the server
$z^{i,i'}_{j,k,j',k'}$	a binary integer variable indicating whether the data transmission from tasks j to k of application i precedes another data transmission from tasks j' to k' of application i'
s	channel transmission rate
I	a positive constant that is larger than any value we use for a variable in this paper
δ^i	execution start time of application i

$$y^i_{j,k} \leq x^i_j + x^i_k,\ j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \quad (8)$$

$$y^i_{j,k} \leq 2 - x^i_j - x^i_k,\quad j,k \in \mathcal{N}^i, e^i_{j,k} \in \mathcal{E}^i, i \in \mathcal{N} \quad (9)$$

$$0 \leq (\omega^i_{j,k} - \omega^{i'}_{j',k'})/I + z^{i,i'}_{j,k,j',k'}, j,k \in \mathcal{N}^i, j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}, (i,j,k) \neq (i',j',k') \quad (10)$$

$$1 \geq (\omega^i_{j,k} - \omega^{i'}_{j',k'})/I + z^{i,i'}_{j,k,j',k'},\ j,k \in \mathcal{N}^i, j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}, (i,j,k) \neq (i',j',k') \quad (11)$$

$$1 = z^{i,i'}_{j,k,j',k'} + z^{i',i}_{j',k',j,k},\ j,k \in \mathcal{N}^i, j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}, (i,j,k) \neq (i',j',k') \quad (12)$$

$$\omega^{i'}_{j',k'} \geq W^i_{j,k} - I(3 - z^{i,i'}_{j,k,j',k'} - y^i_{j,k} - y^{i'}_{j',k'}), j,k \in \mathcal{N}^i, j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}, (i,j,k) \neq (i',j',k') \quad (13)$$

$$x^i_0 = 0, x^i_{M^i} = 0, \tau^i_0 = \delta^i,\ i \in \mathcal{N} \quad (14)$$

Constraint (2) shows the relationship between the start and end times for data transmission from task j to task k of application i. Here, $d^i_{j,k}/s$ denotes the time for data transmission from task j to task k in application i over the shared channel. Constraint (3) shows the relationship between computing start and end times of task j of application i. Constraint (4) shows that the data transmission form task j to task k of application i should be after than the end time of the computing for task j of application i. Constraint (5) indicates that task k of application i can only be processed after receiving all the input data from its previous tasks j ($e^i_{j,k} \in \mathcal{E}^i$). Constraints (6)–(9) show that the value of $y^i_{j,k}$ is determined by the offloading decision variables x^i_j and x^i_k. This binary integer variable indicates whether the data sent from task j to k is transmitted over the communication channel or not. If both of tasks j and k are decided to be offloaded, the value of $y^i_{j,k}$ is 0. On the other hand, if only one of either task j or k is decided to be offloaded, the value of $y^i_{j,k}$ will be 1. Constraints (10)–(12) show that the value of $z^{i,i'}_{j,k,j',k'}$ is determined by the transmission start times for applications i and i', $\omega^i_{j,k}$ and $\omega^{i'}_{j',k'}$, respectively. Here, a positive constant denoted by I is used and it is greater than any other variable. We have $z^{i,i'}_{j,k,j',k'} = 1$ if the data transmission from task j to k in application i is early than the data transmission from task j' to k' in application i'. Constraint (13) guarantees that there is only one data transmission over the communication channel at any time. For two data transmissions, i.e., $y^i_{j,k} = y^{i'}_{j',k'} = 1$, if the transmission from task j to k in application i is earlier than that from task j' to k' in application i', i.e., $z^{i,i'}_{j,k,j',k'} = 1$, $\omega^{i'}_{j',k'}$ should be later than the end time $W^i_{j,k}$ to avoid the collisions of the two transmissions.

When the number of applications in the system and the number of tasks in each application are limited, we can utilize an appropriate optimization software package such as the Gurobi Optimizer [10] to solve the static offloading problem. However, in real systems, it is difficult to know when an application is executed in advance. In the next section, we focus on how to determine the offloading decision for each application on condition that the application can be executed at any time.

V. DYNAMIC OFFLOADING PROBLEM

Generally, each application is executed repeatedly but without any information about when it is executed. In this paper, we consider the case in which each application can be executed at any time at a mobile device and after the execution the device sends the information about the application such as the execution time of each task and the data size sent between tasks to the edge server to make the offloading decision. The server then sends the offloading decision back to the mobile device. Therefore, the time needed for the offloading decision is also crucially important to the offloading approach. Since the data size of the characteristics and the offloading decision for an application is significantly smaller than that sent between tasks of that application, we assume that the delay for transmitting the information about an application and the offloading decision can be ignored.

For a newly executed application, its data transmission cannot interfere with the underway data transmissions of other applications that are executed earlier than the new application.

However, the offloading decisions of those tasks and data transmissions that are not still be executed or transmitted may be changed in order to minimize the average application completion time. The set of applications including those under execution and the newly executed one is denoted by $\mathcal{N}'$, but the applications are ordered by their execution start times. Especially, the newly executed application is denoted by N', and we let $\mathcal{N}' = \{1, 2, \cdots, N'\}$. For each executed application i $(i < N')$ we can have the following relations (15)–(20).

$$\begin{aligned} x'^i_j &= x^i_j, i < N', j \in \mathcal{N}^i && (15)\\ \tau'^i_k &= \tau^i_k, i < N', j \in \mathcal{N}^i && (16)\\ T'^i_j &= T^i_j, i < N', j \in \mathcal{N}^i && (17)\\ y'^i_{j,k} &= y^i_{j,k}, i < N', j,k \in \mathcal{N}^i && (18)\\ \omega'^i_{j,k} &= \omega^i_{j,k}, i < N', j,k \in \mathcal{N}^i && (19)\\ W'^i_{j,k} &= W^i_{j,k}, i < N', j,k \in \mathcal{N}^i && (20) \end{aligned}$$

Here, we attempt to minimize the completion time of the newly executed application but at the same time to avoid the disturbance to the applications under execution. Therefore, we reconsider the offloading decisions only for the tasks and the data transmissions of the executed applications that are not still be performed. We formulate a new MIP problem (21) for each newly executed application N' by extending problem (1) as follows.

$$\min \quad T = \frac{1}{N'} \sum_{i \in \mathcal{N}'} \left(T^i_{M^i} - \tau^i_0\right), \qquad (21)$$

subject to

$$\begin{aligned} & (2)\text{–}(14) \\ x^i_j &= x'^i_j, \tau^i_k = \tau'^i_k, T^i_j = T'^i_j, i < N', \\ & \qquad j \in \mathcal{N}^i, \tau'^i_k \le \delta^{N'}, && (22)\\ y^i_{j,k} &= y'^i_{j,k}, \omega^i_{j,k} = \omega'^i_{j,k}, W^i_{j,k} = W'^i_{j,k}, \\ & \qquad i < N', j,k \in \mathcal{N}^i, \omega'^i_{j,k} \le \delta^{N'}. && (23) \end{aligned}$$

Besides constraints (2)–(14) of problem (1), we have two additional constraints (22) and (23) to guarantee not to change the underway task offloading and data transmission decisions of previous applications.

We can solve problem (21) to obtain the offloading decision for each newly executed application. However, when the network becomes congested, that is, there are many transmission requests to the shared communication network, the computation time for solving problem (21) may become negligibly long, significantly poisoning the application completion time. By a simulation experiment in which there are 50 applications and the inter-execution of each application is only 40s, we obtained a unrealistic result showing that the average computation time is larger than 1000s. Therefore, this approach is difficult to apply in real systems.

In order to reduce the computation time, we do not touch any offloading decision of previously executed applications. Then, the offloading decision problem for a newly executed application can be formulated as the following MIP problem (24).

$$\begin{aligned} \min \quad T &= T^{N'}_{M^{N'}} - \tau^{N'}_0 && (24)\\ \text{subject to} & \\ W^{N'}_{j,k} &= \omega^{N'}_{j,k} + y^{N'}_{j,k} d^{N'}_{j,k}/s, \\ & \qquad j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (25)\\ T^{N'}_j &= \tau^{N'}_j + x^{N'}_j c^{N'}_j + (1 - x^{N'}_j) m^{N'}_j, \\ & \qquad j \in \mathcal{N}^{N'} && (26)\\ \omega^{N'}_{j,k} &\ge T^{N'}_j, \ j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (27)\\ \tau^{N'}_k &\ge W^{N'}_{j,k}, \ j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (28)\\ y^{N'}_{j,k} &\ge x^{N'}_j - x^{N'}_k, \ j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (29)\\ y^{N'}_{j,k} &\ge x^{N'}_k - x^{N'}_j, \ j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (30)\\ y^{N'}_{j,k} &\le x^{N'}_j + x^{N'}_k, \ j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (31)\\ y^{N'}_{j,k} &\le 2 - x^{N'}_j - x^{N'}_k, \\ & \qquad j,k \in \mathcal{N}^{N'}, e^{N'}_{j,k} \in \mathcal{E}^{N'} && (32)\\ 0 &\le (\omega^i_{j,k} - \omega^{i'}_{j',k'})/I + z^{i,i'}_{j,k,j',k'}, \ j,k \in \mathcal{N}^i, \\ & \qquad j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, \\ & \qquad i,i' \in \mathcal{N}', (i,j,k) \ne (i',j',k') && (33)\\ 1 &\ge (\omega^i_{j,k} - \omega^{i'}_{j',k'})/I + z^{i,i'}_{j,k,j',k'}, \ j,k \in \mathcal{N}^i, \\ & \qquad j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, \\ & \qquad i,i' \in \mathcal{N}', (i,j,k) \ne (i',j',k') && (34)\\ 1 &= z^{i,i'}_{j,k,j',k'} + z^{i',i}_{j',k',j,k}, \ j,k \in \mathcal{N}^i, \\ & \qquad j',k' \in \mathcal{N}^{i'}, e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, \\ & \qquad i,i' \in \mathcal{N}', (i,j,k) \ne (i',j',k') && (35)\\ \omega^{i'}_{j',k'} &\ge W^i_{j,k} - I(3 - z^{i,i'}_{j,k,j',k'} - y^i_{j,k} - y^{i'}_{j',k'}), \\ & \qquad j,k \in \mathcal{N}^i, j',k' \in \mathcal{N}^{i'}, \\ & \qquad e^i_{j,k} \in \mathcal{E}^i, e^{i'}_{j',k'} \in \mathcal{E}^{i'}, i,i' \in \mathcal{N}', \\ & \qquad (i,j,k) \ne (i',j',k') && (36)\\ x^{N'}_0 &= 0, x^{N'}_{M^{N'}} = 0, \tau^{N'}_0 = \delta^{N'} && (37)\\ y^i_{j,k} &= y'^i_{j,k}, \omega^i_{j,k} = \omega'^i_{j,k}, W^i_{j,k} = W'^i_{j,k}, \\ & \qquad i < N', j,k \in \mathcal{N}^i, \omega'^i_{j,k} \le \delta^{N'} && (38) \end{aligned}$$

Constraint (38) guarantees that the offloading decisions of the previous applications will be unchanged. Since the number of decision variables here is much less than that of problem (21), the computation complexity become smaller.

VI. Performance Evaluation

In this section, we describe the performance evaluation of our proposed approach in comparison with a previous algorithm that is proposed by Wu *et al.* [9], and is shown by *"Partial offloaded"* in the figures. Two extreme approaches, called *"Full-offloaded"* and *"No offloaded"*, respectively, were also simulated for comparison and in the former all the tasks except task 0 and M^i of each application are offloaded while in the latter each application is executed locally at the mobile device it arrives. We considered a system model in which there are 50 mobile devices each of which has the same computation power and each mobile device executes its mobile application 20 times with different input data. We assumed that each application contains 20 tasks and the task flow graph was randomly generated. The computation time of each task locally at a mobile device was generated randomly between $[4, 12]$s and the computation times of tasks m_0^i and $m_{M^i}^i$ were ignored. We also assumed that the computation power of each mobile device is the same and the ratio of computation speed of the server to a mobile device was fixed to be 4. The time instant at which an application is executed at a mobile device was generated randomly following the Poisson process, i.e., the inter-execution time between applications follows the exponential distribution. The data size sent between tasks was generated randomly between $[10, 40]$MB.

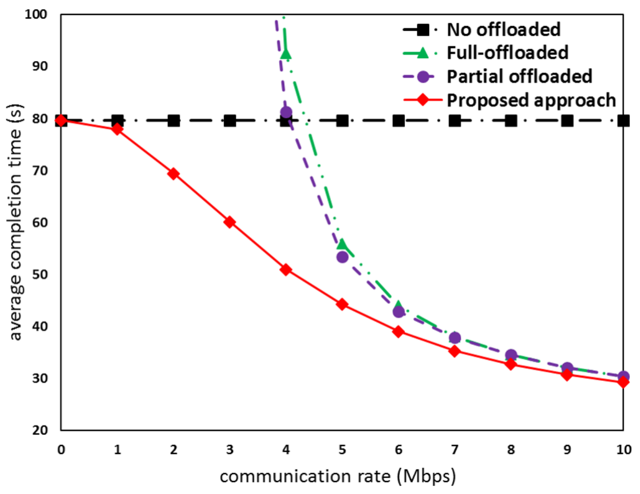

Fig. 4. Effect of communication speed.

A. *Effect of communication speed*

Figure 4 shows the average application completion time of various algorithms when changing the speed of the shared communication channel. The application completion time of our proposed approach in Figure 4 include the computation time of our approach. The average inter-execution time between two adjacent application execution is 40s. From Figure 4, we can see that our proposed approach performs much better than all the other algorithms for a wide range of communication speed.

When the communication speed is extremely low it is not beneficial to offload any task to the server due to the communication delay, and we can see that our proposed approach performs similarly to the case of ”No offloaded”. Under low communication speed, although a lot of application processing are overlap with each others, which will lead to longer computation time of our proposed approach, the performance of our proposed approach is still better other algorithm. When the communication speed is very fast, offloading tasks to the server leads to large improvement over the application completion time and we can see in this case that our proposed approach performs similarly to the case of ”Full offloaded”. However, even though the communication speed is very fast, there still are some tasks need not to be offloaded and our proposed approach can accurately identify these tasks. Furthermore, the computation time of our proposed approach was small enough to be ignored in this case.

We also see that our proposed approach outperforms the ”Partial offloaded” algorithm, even though we assumed that the ”Partial offloaded” algorithm always knows the transmission delay exactly in order not to cause any bias to its offloading decisions. The key drawback of the ”Partial offloaded” algorithm is that possible collisions between data transmissions over the shared communication network are not taken into account, leading to worse performance when the communication speed becomes slower.

B. *Effect of application inter-execution time*

Figure 5 shows the average application completion time of the algorithms under consideration when changing the application inter-execution time. The shared channel speed was fixed to 5Mbps. From Figure 5, we see that our proposed approach performs much better than other ones except when the network is very congested, e.g., when the inter-execution time is 5s. That means when the network is very congested, we can just turn off the offloading mechanism and just let each mobile device process the arriving application.

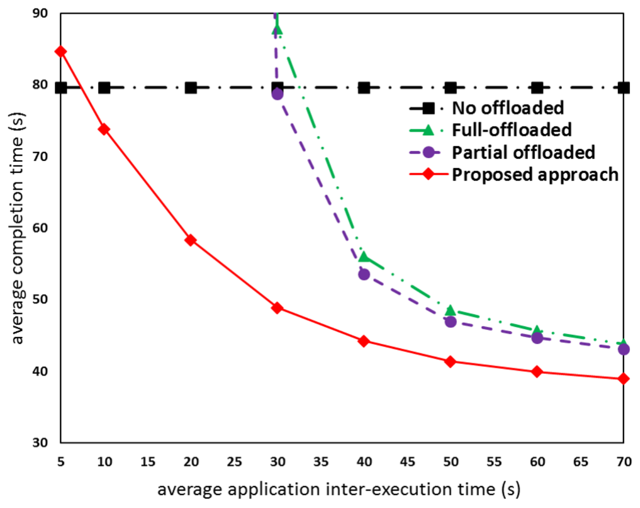

Fig. 5. Effect of application inter-execution time.

We also show the average computation time for solving problem (24) with 5Mbps communication rate and 10Mbps communication rate respectively in Figure 6. From this figure, we can see that the solution computation time becomes long when the network becomes congested. For example, when the inter-execution time is 5s, the computation time is 8s leading the server to be overloaded. When the communication speed becomes faster, there are fewer collisions leading the average computation time to be faster.

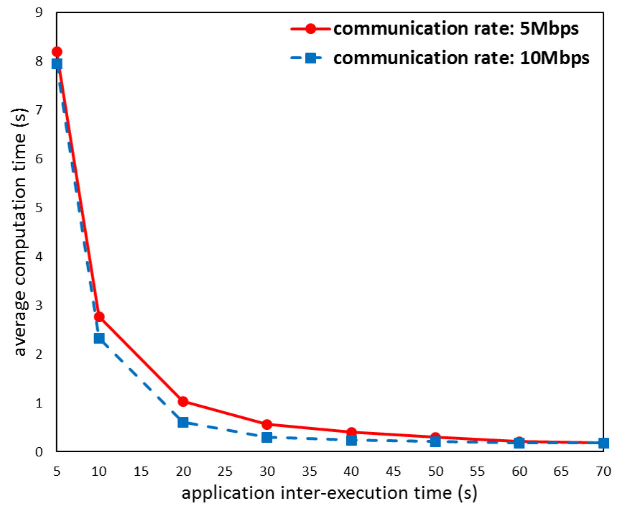

Fig. 6. Average computation time of our proposed approach.

C. Effect of number of tasks of an application

The number of tasks of an application will affect the computation time of our proposed approach. Hence, we also compared the effect of different number of tasks of an application. We assumed that there is only one mobile device and executed its application only once. In our simulations, we considered the number of tasks of an application are 10, 20, 30, 40, 50 respectively and compared the computation time of our proposed approach. In each simulation, we executed the application 1000 times with different task flow and input data and calculated the average computation time of our approach. From Figure 7, we can find that it has been growing fast when the number of tasks of an application is greeter than 40. On the other hand, our approach performs very well when the number of tasks of an application is less than 40.

VII. Conclusion

In this paper, we focused on the problem of how to offload the computation intensive tasks of newly executed mobile applications in order to minimize the average completion time of all the applications. We considered a realistic model for each application in which tasks are expressed by a task flow graph. We also considered the collisions at shared communication network when more than one mobile device attempts to transmit data simultaneously. At first, we formulated the static offloading problem as a mixed integer programming (MIP) problem. Then, we extended the static problem to a dynamic offloading problem and solved this problem by optimization software packages. The proposed approach was examined by simulation experiments and the results show that our proposed approach outperforms significantly previous ones in a wide range of system parameters. Furthermore, the results also show that our approach is efficient when the number of tasks in each application are limited.

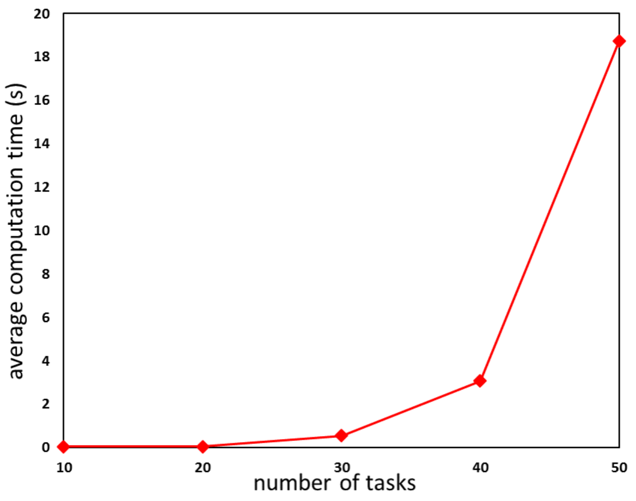

Fig. 7. Effect of number of tasks in an application.

References

[1] X. Fan, J. Cao, and H. Mao, ”A survey of mobile cloud computing”, *ZTE Communications*, Vol. 9, No. 1, 2010; 4–8.

[2] F. Liu, *et al.* ”Gearing Resource-Poor Mobile Devices with Powerful Clouds: Architecture, Challenges and Applications”, *IEEE Wireless Communications*, Vol. 20, No. 3, 2013; 14–22.

[3] K. Kumar, and Y. Lu, ”Cloud computing for mobile users: Can offloading computation save energy?”, *IEEE Computer* Vol.43, No.4, 2010; 51–56.

[4] B. Chun, *et al.* ”Clonecloud: elastic execution between mobile device and cloud”, *Proc. ACM Int. Conf. Computer Systems (EuroSys2011)*, 2011; 301–314.

[5] Z. Li, C. Wang, and R. Xu, ”Computation offloading to save energy on handheld devices: a partition scheme”, *Proc. ACM Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems (CASES2001)*, 2001; 238–246.

[6] M. V. Barbera, *et al.* ”To offload or not to offload? the bandwidth and energy costs of mobile cloud computing”, *Proc. IEEE Int. Conf. Computer Communications (INFOCOM2013)*, 2013; 1285–1293. 10

[7] M. R. Ra, *et al.* ”Odessa: enabling interactive perception applications on mobile devices”, *Proc. ACM 9th Int. Conf. Mobile Systems, Applications, and Services (MobiSys2011)*, 2011; 43–56.

[8] M. Jia, J. Cao, and L. Yang, ”Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing”, *Proc. IEEE Int. Conf. Computer Communications Workshops (INFOCOM WKSHPS)*, 2014; 352–357.

[9] H. Wu, W. Knottenbelt, K. Wolter, and Y. Sun, ”An Optimal Offloading Partitioning Algorithm in Mobile Cloud Computing”, *Int. Conf. Quantitative Evaluation of Systems (QEST2016)* in LNCS 9826, eds. G. Agha and B.V. Houdt, Springer, 2016; 311–328.

[10] Gurobi Optimization, I. ”Gurobi Optimizer Reference Manual (2016)”, http://www.gurobi.com/, posted on 2017.

[11] Spacey, S. ”Concise CPLEX”, *Imperial Technical Paper* http://www. doc. ic. ac. uk/research/technicalreports/2009/DTR09-7. pdf/, posted on 2009.

[12] Trummer I, Koch C. ”Solving the Join Ordering Problem via Mixed Integer Linear Programming.” *Proc. ACM Int. Conf. Management of Data. (SIGMOD 2017)*, 2017; 1025–1040.

[13] A. B. Craig, ”Understanding Augmented Reality: concepts and applications”, 1st Ed., Morgan Kaufmann, 2013.

[14] X. Chen, ”Decentralized computation offloading game for mobile cloud computing”, *IEEE Trans. Parallel and Distributed Systems*, Vol. 26, No. 4, 2015; 974–983.

[15] X. Chen, *et al*. ”Efficient multi-user computation offloading for mobile-edge cloud computing”, *IEEE/ACM Trans. Networking*, 2015; 1–14.

[16] E. Meskar, *et al*. ”Energy efficient offloading for competing users on a shared communication channel”, *Proc. IEEE Int. Conf. Communications (ICC2015)*, 2015; 3192–3197.

[17] L. Yang, *et al*. ”Multi-user computation partitioning for latency sensitive mobile cloud applications”, *IEEE Trans. Computers*, Vol. 64, No. 8, 2015; 2253–2266.