Design and Analysis of Heuristic Algorithms for Energy-Constrained Task Scheduling With Device-Edge-Cloud Fusion

Keqin Li, *Fellow, IEEE*

Abstract—Mobile edge computing with device-edge-cloud fusion provides a new type of heterogeneous computing environment. We consider task scheduling with device-edge-cloud fusion (without energy concern) and energy-constrained task scheduling with device-edge-cloud fusion as combinatorial optimization problems. The main contributions of the paper are summarized as follows. We design three heuristic algorithms for task scheduling with device-edge-cloud fusion and prove an asymptotic performance bound. We design one heuristic algorithm for energy-constrained task scheduling with device-edge-cloud fusion, which solves the two subproblems of task scheduling and power allocation in an interleaved way. We derive lower bounds for the optimal solutions for both task scheduling with device-edge-cloud fusion and energy-constrained task scheduling with device-edge-cloud fusion, so that the performance of our heuristic algorithms can be compared with that of an optimal algorithm. We experimentally evaluate the performance of our heuristic algorithms and find that the performance of our heuristic algorithms are very close to that of optimal algorithms. To the best of our knowledge, this is the first paper which studies task scheduling with device-edge-cloud fusion and energy-constrained task scheduling with device-edge-cloud fusion as combinatorial optimization problems and conducts analytical performance evaluation.

Index Terms—Absolute performance bound, asymptotic performance bound, device-edge-cloud fusion, energy-constrained task scheduling, heuristic algorithm, mobile edge computing

1 Introduction

1.1 Background and Motivation

Mobile edge computing with device-edge-cloud (D-E-C) fusion provides a new type of heterogeneous computing environment. Such device, edge, and cloud collaborative processing is able to integrate various computing and communication resources from mobile devices, edge servers, and cloud servers for high-speed execution of real applications. Device-edge-cloud collaborative computing has been an very active research area in recent years (see [9], [25] for some comprehensive surveys). Various applications have been supported by device-edge-cloud collaborative computing, such as augmented reality [30], Big Data [1], [8], cognitive service [5], deep learning [16], IoT [26], and video query [28]. However, such an environment has heterogeneous computing capabilities as well as sophisticated wireless communication and wired (Internet) communication costs, which makes efficient mobile edge computing with D-E-C fusion very challenging.

There are two main considerations in mobile edge computing, i.e., performance and power. Both considerations are from a mobile device's point of view. For performance, we are interested in finishing a set of tasks generated on a mobile device as soon as possible, with the assistance of an edge server and a cloud server. For power, we are only interested in energy consumption for computation and communication of the mobile device, not the edge server and the cloud server.

There are two main considerations in task scheduling, i.e., execution time and energy consumption. Execution time includes computation time and communication time. Typically, an edge server has faster computation speed than a mobile device, and a cloud server has even faster computation speed. However, offloading a task from a mobile device to an edge server incurs wireless communication time, and offloading a task to a cloud server incurs both wireless communication time and wired communication time. Such communication cost and delay offset the gain from faster computing speeds, and make execution time minimization in mobile edge computing with D-E-C fusion more challenging than in other systems and environments. A mobile device can minimize its energy consumption for computation and communication by adjusting its computation speed and wireless communication speed. However, energy consumption for computation and communication of an edge server and a cloud server is not included into consideration. The computation speeds of the servers and the Internet communication speed are not controllable by a mobile device.

There are two main challenges in studying heuristic algorithms for energy-constrained task scheduling with device-edge-cloud fusion. The first challenge is to design the strategies in a heuristic algorithm. An effective heuristic algorithm should be able to complete a set of tasks in the shortest period of time within given energy budget. The

• *The author is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 4 July 2022; revised 9 October 2022; accepted 18 October 2022. Date of publication 25 October 2022; date of current version 8 June 2023.
Recommended for acceptance by P. D. Yoo.
Digital Object Identifier no. 10.1109/TSUSC.2022.3217014

2377-3782 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

second challenge is to analyze the performance of a heuristic algorithm. An efficient heuristic algorithm should be evaluated in such a way that its performance is compared with that of an optimal algorithm.

The motivation of this paper is to study task scheduling in mobile edge computing with device-edge-cloud fusion. We consider task scheduling with device-edge-cloud fusion (without energy concern) and energy-constrained task scheduling with device-edge-cloud fusion as combinatorial optimization problems. Both are NP-hard problems. We have two main objectives, i.e., (1) to design efficient heuristic algorithms with/without energy constraint; (2) and to conduct analytical and experimental performance evaluation. One strong and unique feature of our investigation is to derive lower bounds for optimal solutions and to compare heuristic solutions with optimal solutions.

1.2 Related Research

Task offloading and execution in mobile edge computing with device-edge-cloud fusion has been considered by several researchers. Ding et al. formulated a potential game in which each user end selfishly minimizes its weighted sum of energy consumption and time consumption for both hierarchical and horizontal end-edge-cloud computing [6]. Du et al. minimized the total cost (computation cost + communication cost + transferring cost) of all tasks by making decisions on task transferring among edge and cloud servers [7]. Fantacci and Picano minimized the mean overall response time of edge computing servers and the number of unsatisfied application flows (whose deadlines are missed) [10]. Feng et al. minimized a weighted sum of total execution time and total energy consumption of a smart device, an edge server, and a cloud server by proper decision on local accuracy, subcarrier assignment, CPU frequency, and transmission power [11]. Guo et al. investigated energy-efficient and delay-guaranteed workload allocation in an IoT-edge-cloud computing system by minimizing energy consumption while provisioning delay guarantee for end users [13]. Hao et al. proposed an intelligent task offloading scheme for a cloud-edge collaborative system by considering coarse-grained computing and fine-grained computing [14]. Huang et al. minimized a weighted sum of energy consumption and execution time for each task, for which there is a task offloading decision and a maximum tolerable delay [15]. Liu et al. minimized the total computation cost and the total routing cost by workflow assignment and traffic routing [21]. Peng et al. minimized the total processing time and the total energy consumption, and maximized the average resource utilization of edge servers [23]. Ren et al. optimized a weighted average service delay of all mobile devices, taking into account wireless transmission delay of mobile devices, backhaul transmission delay of the edge server, computation delay of the edge server, and computation delay of the cloud server [24]. Sun et al. minimized the total weighted sum of normalized execution time (transmission time + computation time) and normalized energy consumption of all mobile devices [27].

However, there are two main limitations in the current literature. First, task scheduling in a device, edge, and cloud collaborative computing environment has never been studied in the context of combinatorial optimization. For instance, the most important metrics in scheduling, i.e., the makespan or the schedule length, has not been considered by any researcher in mobile edge computing with device-edge-cloud fusion. It should be emphasized that minimizing the sum of individual execution times of all tasks is entirely different from minimizing the overall execution time of all tasks (i.e., the makespan). Second, minimizing a weighted sum of execution time and energy consumption is entirely different from minimizing the makespan with certain energy constraint. Furthermore, since execution time and energy consumption have totally different measures (the former is measured in seconds, and the latter is measured in Joule), a weighted sum of the two quantities has inherent technical flaw and makes little sense.

Researchers have also investigated resource allocation, scaling, sharing, and management [4], [17], [20], [31], as well as pricing and profit related issues [2], [22], [29] in mobile edge computing environments with device-edge-cloud fusion.

1.3 New Contributions

The main contributions of the paper are summarized as follows.

- We design three heuristic algorithms for task scheduling with device-edge-cloud fusion and prove an asymptotic performance bound.
- We design one heuristic algorithm for energy-constrained task scheduling with device-edge-cloud fusion, which solves the two subproblems of task scheduling and power allocation in an interleaved way.
- We derive lower bounds for the optimal solutions for both task scheduling with device-edge-cloud fusion and energy-constrained task scheduling with device-edge-cloud fusion, so that the performance of our heuristic algorithms can be compared with that of an optimal algorithm.
- We experimentally evaluate the performance of our heuristic algorithms and find that the performance of our heuristic algorithms are very close to that of optimal algorithms.

To the best of our knowledge, this is the first paper which studies task scheduling with device-edge-cloud fusion and energy-constrained task scheduling with device-edge-cloud fusion as combinatorial optimization problems and conducts analytical performance evaluation.

The rest of the paper is organized as follows. In Section 2, we present our models and problems. In Section 3, we consider the problem of task scheduling with device-edge-cloud fusion. In Section 4, we consider the problem of energy-constrained task scheduling with device-edge-cloud fusion. We conclude the paper in Section 5.

2 Models and Problems

In this section, we present our models and problems.

2.1 Models

In this section, we describe our task model, computation and communication models, and power consumption models.

TABLE 1
Notations and Definitions

Symbol	Definition
S	$= \{t_1, t_2, \ldots, t_n\}$, a set of independent tasks
n	the number of tasks
t_i	$= (d_i, r_i)$, a task
d_i	the amount of communication of t_i
r_i	the amount of computation of t_i
D	the total amount of communication
R	the total amount of computation
(S_d, S_e, S_c)	a schedule
S_d	the set of task processed on the mobile device
S_e	the set of task processed on the edge server
S_c	the set of task processed on the cloud server
s_d	the computation speed of the mobile device
s_e	the computation speed of the the edge server
s_c	the computation speed of the the cloud server
$\tilde{s}_e$	the communication speed between the mobile device and the edge server
$\tilde{s}_c$	the communication speed between the edge server and the cloud server
τ_i	the execution time of task t_i
D_d	the total amount of communication of tasks in S_d
R_d	the total amount of computation of tasks in S_d
T_d	the total execution time of tasks in S_d
D_e	the total amount of communication of tasks in S_e
R_e	the total amount of computation of tasks in S_e
T_e	the total execution time of tasks in S_e
D_c	the total amount of communication of tasks in S_c
R_c	the total amount of computation of tasks in S_c
T_c	the total execution time of tasks in S_c
P	the power consumption for computation
P_d	the dynamic component of power consumption
P_s	the static component of power consumption
ξ, α	parameters of the power consumption model
P_t	the transmission power
w	the communication bandwidth
β	communication channel property
T	$= \max\{T_d, T_e, T_c\}$, the total execution time (i.e., the schedule length)
E	the total energy consumption
$\hat{E}$	energy constraint
γ_i	$= d_i/r_i$, the communication-to-computation ratio
γ'	$= \min\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$
γ''	$= \max\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$
$T^*(S)$	the optimal schedule length
$T(S)$	the heuristic schedule length
τ	the maximum execution time of any task anywhere

Table 1 lists all the symbols and their definitions in the order introduced in this paper.

We consider a fog computing environment with one mobile device, one edge server, and one cloud server (see Fig. 1). When a task is generated on the mobile device, the mobile device can process the task locally on the device, or offload the task to the edge server for remote processing. The edge server can further send the task to the cloud server for processing. The mobile device communicates with the edge server via wireless communication and the edge server communicates with the cloud server via the Internet.

Assume that there is a set S of n independent tasks $S = \{t_1, t_2, \ldots, t_n\}$. A task is specified as $t_i = (d_i, r_i)$, where d_i is the amount of communication (measured by the number of million (or mega) bits (MB)), and r_i is the amount of computation

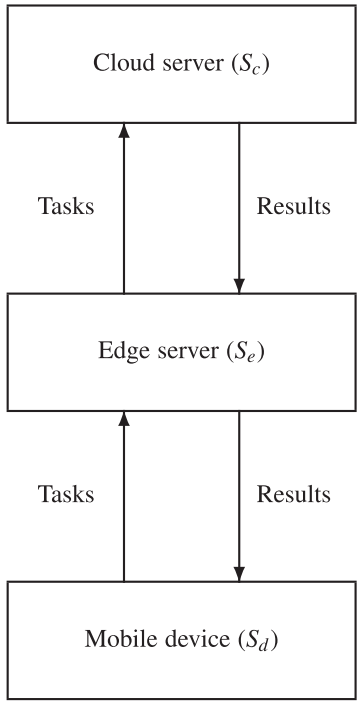

Fig. 1. Task scheduling with device-edge-cloud fusion.

(measured by the number of billion instructions (BI) or giga instructions (GI)). We define

$$D = \sum_{t_i \in S} d_i,$$

to be the total amount of communication, and

$$R = \sum_{t_i \in S} r_i,$$

to be the total amount of computation.

A *schedule* is a partition of S into three disjoint subsets: (S_d, S_e, S_c), such that

$$S = S_d \cup S_e \cup S_c,$$

where S_d is the set of task processed on the mobile device, S_e is the set of task processed on the edge server, and S_c is the set of task processed on the cloud server.

It is already known that for energy-constrained task scheduling, the total execution time on a mobile device is minimized when all tasks have the same computation speed s_d on the device, and the total execution time on an edge server is minimized when all tasks have the same communication speed $\tilde{s}_e$ between the mobile device and the edge server [19]. Therefore, we assume that the mobile device has computation speed s_d for all tasks, the edge server has computation speed s_e for all tasks, and the cloud server has computation speed s_c for all tasks. All computation speeds are measured by billion instructions per second (BI/s). Furthermore, the wireless communication speed between the mobile device and the edge server is $\tilde{s}_e$ for all tasks, and the Internet communication speed between the edge server and the cloud server is $\tilde{s}_c$ for all tasks. All communication speeds are measured by million bits per second (Mbps).

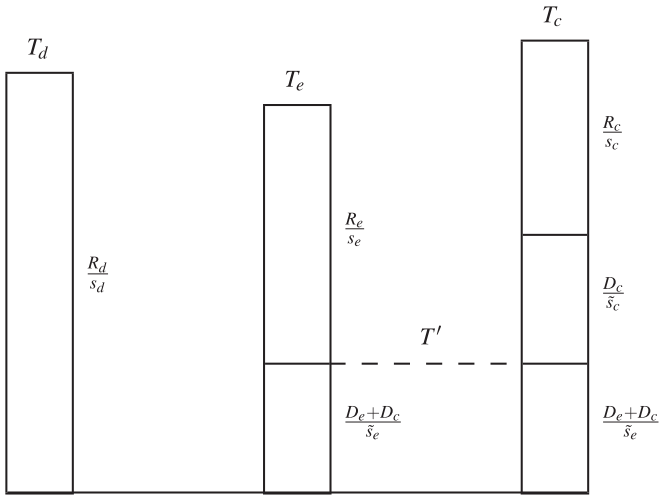

Fig. 2. Illustration of T_d, T_e, T_c.

Let τ_i be the execution time (measured by seconds) of task t_i. There are three execution modes (see Fig. 2).

- Device computing – The first execution mode is to process a task on the mobile device, which gives

$$\tau_i = \frac{r_i}{s_d}.$$

Let us define

$$D_d = \sum_{t_i \in S_d} d_i,$$

to be the total amount of communication of tasks in S_d, and

$$R_d = \sum_{t_i \in S_d} r_i,$$

to be the total amount of computation of tasks in S_d. The total execution time of tasks in S_d is

$$T_d = \sum_{t_i \in S_d} \tau_i = \sum_{t_i \in S_d} \frac{r_i}{s_d} = \frac{R_d}{s_d}.$$

Note that T_d has only one component, i.e., mobile device computation time, which is controllable by the mobile device by changing s_d.

- Edge computing – The second execution mode is to process a task on the edge server, which gives

$$\tau_i = \frac{d_i}{\tilde{s}_e} + \frac{r_i}{s_e}.$$

Let us define

$$D_e = \sum_{t_i \in S_e} d_i,$$

to be the total amount of communication of tasks in S_e, and

$$R_e = \sum_{t_i \in S_e} r_i,$$

to be the total amount of computation of tasks in S_e. The total execution time of tasks in S_e is

$$T_e = \sum_{t_i \in S_e} \tau_i + \frac{D_c}{\tilde{s}_e} = \sum_{t_i \in S_e} \left(\frac{d_i}{\tilde{s}_e} + \frac{r_i}{s_e} \right) + \frac{D_c}{\tilde{s}_e} = \frac{D_e + D_c}{\tilde{s}_e} + \frac{R_e}{s_e},$$

where we notice that the edge server is also responsible for wireless communication of tasks in S_c with the definition of D_c to be given shortly. Note that T_e has two components, i.e., wireless communication time (which is controllable by the mobile device by changing $\tilde{s}_e$) and edge server computation time.

- Cloud computing – The third execution mode is to process a task on the cloud server, which gives

$$\tau_i = \frac{d_i}{\tilde{s}_e} + \frac{d_i}{\tilde{s}_c} + \frac{r_i}{s_c}.$$

Let us define

$$D_c = \sum_{t_i \in S_c} d_i,$$

to be the total amount of communication of tasks in S_c, and

$$R_c = \sum_{t_i \in S_c} r_i,$$

to be the total amount of computation of tasks in S_c. The total execution time of tasks in S_c is

$$T_c = \sum_{t_i \in S_c} \tau_i + \frac{D_e}{\tilde{s}_e} = \sum_{t_i \in S_c} \left(\frac{d_i}{\tilde{s}_e} + \frac{d_i}{\tilde{s}_c} + \frac{r_i}{s_c} \right) + \frac{D_e}{\tilde{s}_e} = \frac{D_e + D_c}{\tilde{s}_e} + \frac{D_c}{\tilde{s}_c} + \frac{R_c}{s_c},$$

where we notice that the cloud server relies on the edge server for wireless communication and must wait for the wireless communication time of tasks in S_e. Note that T_c has three components, i.e., wireless communication time (which is controllable by the mobile device by changing $\tilde{s}_e$), Internet communication time, and cloud server computation time.

Fig. 2 illustrates the above three execution modes. Tasks in S_d are executed locally on the mobile device. Tasks in $S_e \cup S_c$ are transmitted to the edge server via wireless communication. Tasks in S_e are executed on the edge server. Tasks in S_c are further transmitted to the cloud server via Internet communication and executed on the cloud server [6], [24], [27].

The power consumption P (measured by Watts) for computation of the mobile device is

$$P = P_d + P_s = \xi s_d^{\alpha} + P_s,$$

where $P_d = \xi s_d^{\alpha}$ is the dynamic component of power consumption, P_s is the static component of power consumption, and ξ, α are technology-dependent constants [18], [19].

The power consumption for communication and the wireless communication speed is related as

$$\tilde{s}_e = w \log_2(1 + \beta P_t),$$

and

$$P_t = \frac{2^{\tilde{s}_e/w} - 1}{\beta},$$

where P_t is the transmission power (measured by Watts), w is the communication bandwidth (measured by Mbps), and β is a quantity determined by the communication channel [18], [19].

2.2 Problems

In this section, we define our combinatorial optimization problems and prove their NP-hardness.

In mobile edge computing, both power and performance are considered from the mobile device's point of view. The total execution time (i.e., the makespan, or the schedule length, measured by seconds) is

$$T = \max\{T_d, T_e, T_c\}.$$

The energy consumption for computation of the mobile device (measured by Joule) is

$$PT_d = (\xi s_d^\alpha + P_s)\frac{R_d}{s_d} = \left(\frac{\xi s_d^\alpha + P_s}{s_d}\right) R_d.$$

The energy consumption for communication of the mobile device (measured by Joule) is

$$P_t\left(\frac{D_e + D_c}{\tilde{s}_e}\right) = \frac{2^{\tilde{s}_e/w} - 1}{\beta} \cdot \frac{D_e + D_c}{\tilde{s}_e} = \left(\frac{2^{\tilde{s}_e/w} - 1}{\beta \tilde{s}_e}\right)(D_e + D_c).$$

The total energy consumption of the mobile device (measured by Joule) is

$$E = \left(\frac{\xi s_d^\alpha + P_s}{s_d}\right) R_d + \left(\frac{2^{\tilde{s}_e/w} - 1}{\beta \tilde{s}_e}\right)(D_e + D_c).$$

The controllable variables are s_d and $\tilde{s}_e$.

In this paper, we address two combinatorial optimization problems.

Problem 1. (*Task Scheduling with Device-Edge-Cloud Fusion*) Given S, $s_d, s_e, s_c, \tilde{s}_e, \tilde{s}_c$, find a schedule (S_d, S_e, S_c), such that T is minimized.

Problem 2. (*Energy-Constrained Task Scheduling with Device-Edge-Cloud Fusion*) Given S, $s_e, s_c, \tilde{s}_c$, and energy constraint $\hat{E}$, find a schedule (S_d, S_e, S_c) and speeds $s_d, \tilde{s}_e$, such that T is minimized and that $E = \hat{E}$.

Both problems are NP-hard even for very special cases.

Theorem 1. *The problem of task scheduling with device-edge-cloud fusion is NP-hard.*

Proof. When $s_d = s_e = s_c$ (i.e., homogeneous computing) and $\tilde{s}_e = \tilde{s}_c = \infty$ (i.e., there is no communication time), the problem becomes the classic *multiprocessor scheduling* problem, which is NP-hard ([12], p. 65). □

Theorem 2. *The problem of energy-constrained task scheduling with device-edge-cloud fusion is NP-hard.*

Proof. When $s_c = \tilde{s}_c = 0$, the cloud server is excluded. If only the mobile device and the edge server are involved, the problem is known to be NP-hard [18]. □

3 Task Scheduling

In this section, we consider the problem of task scheduling with device-edge-cloud fusion.

3.1 Algorithm

In this section, we develop a heuristic algorithm for task scheduling with device-edge-cloud fusion.

Algorithm 1 presents our greedy algorithm to solve the problem of task scheduling with device-edge-cloud fusion.

Algorithm 1. Task Scheduling With Device-Edge-Cloud Fusion

Input: $S = \{t_1, t_2, \ldots, t_n\}$, $s_d, s_e, s_c, \tilde{s}_e, \tilde{s}_c$.
Output: A schedule (S_d, S_e, S_c), such that T is minimized.

$S_d \leftarrow \emptyset$, $S_e \leftarrow \emptyset$, $S_c \leftarrow \emptyset$; (1)
$T_d \leftarrow 0$, $T_e \leftarrow 0$, $T_c \leftarrow 0$; (2)
for ($i = 1$; $i \le n$; i++) **do** (3)
  add t_i to S_d or S_e or S_c,
    where T_d or T_e or T_c is the smallest; (4)
  update T_d or T_e or T_c; (5)
end do (6)

Initially, S_d, S_e, S_c are set to be empty sets. For each task t_i, it is added to one of S_d, S_e, S_c, whichever is the earliest available (ties are broken arbitrarily). The time complexity of Algorithm 1 is $O(n)$, since only constant time is spend for each task.

(Note: If the n tasks are properly ordered, the algorithm may achieve an optimal schedule with the minimum schedule length.)

3.2 Analysis

In this section, we analyze our heuristic algorithm for task scheduling with device-edge-cloud fusion.

3.2.1 Optimal Schedule

One difficulty in analyzing the problem is the independence of the d_i's and the r_i's.

Let $\gamma_i = d_i/r_i$, for all $1 \le i \le n$. γ_i is the *communication-to-computation ratio* (measured by MB/GI). γ_i is the amount of communication to be conducted per unit of computation. $1/\gamma_i$ is the amount of computation to be performed per unit of communication.

We consider a special case: $\gamma_i = \gamma$, $d_i = \gamma r_i$, for all $1 \le i \le n$. The amount of computation is uniformly and linearly proportional to the amount of communication.

The following theorem gives a lower bound for the optimal schedule length $T^*(S)$ of S.

Theorem 3. *If $d_i = \gamma r_i$, for all $1 \le i \le n$, the optimal schedule length $T^*(S)$ has the following lower bound:*

$$T^*(S) \ge BR,$$

where $B = B_1/B_2$, and

$$B_1 = \gamma^2 s_e s_c + \gamma s_e \tilde{s}_c + \gamma \tilde{s}_e s_c + \gamma \tilde{s}_e s_c + \tilde{s}_e \tilde{s}_c,$$

and

$$B_2 = \gamma^2 s_d s_e s_c + \gamma s_d s_e \tilde{s}_c + \gamma s_d \tilde{s}_c s_c + \gamma(s_d + s_e)\tilde{s}_e s_c + (s_d + s_e)\tilde{s}_e \tilde{s}_c + \tilde{s}_e \tilde{s}_c s_c.$$

Proof. It is clear that the optimal schedule cannot be better than the case when

$$T_d = T_e = T_c.$$

For otherwise, we can move some workload from a place with longer execution time to another place with shorter execution time. (Note that here we even ignore the n tasks, and simply assume that the total amount of computation R and the total amount of communication $D = \gamma R$ are continuous variables which can be arbitrarily divided.)

Let us assume that the above condition is true. Since $T_e = T_c$, where

$$T_e = \frac{D_e + D_c}{\tilde{s}_e} + \frac{R_e}{s_e} = \frac{\gamma(R_e + R_c)}{\tilde{s}_e} + \frac{R_e}{s_e},$$

and

$$T_c = \frac{D_e + D_c}{\tilde{s}_e} + \frac{D_c}{\tilde{s}_c} + \frac{R_c}{s_c} = \frac{\gamma(R_e + R_c)}{\tilde{s}_e} + \frac{\gamma R_c}{\tilde{s}_c} + \frac{R_c}{s_c},$$

we have

$$\frac{R_e}{s_e} = \frac{\gamma R_c}{\tilde{s}_c} + \frac{R_c}{s_c} = \left(\frac{\gamma}{\tilde{s}_c} + \frac{1}{s_c}\right) R_c,$$

which implies that

$$R_e = \left(\frac{\gamma s_e}{\tilde{s}_c} + \frac{s_e}{s_c}\right) R_c,$$

and

$$T_e = \frac{\gamma}{\tilde{s}_e}\left(\frac{\gamma s_e}{\tilde{s}_c} + \frac{s_e}{s_c} + 1\right) R_c + \left(\frac{\gamma}{\tilde{s}_c} + \frac{1}{s_c}\right) R_c.$$

Furthermore, since

$$T_d = \frac{R_d}{s_d} = T_e = \left(\frac{\gamma^2 s_e}{\tilde{s}_e \tilde{s}_c} + \frac{\gamma s_e}{\tilde{s}_e s_c} + \frac{\gamma}{\tilde{s}_e} + \frac{\gamma}{\tilde{s}_c} + \frac{1}{s_c}\right) R_c,$$

we have

$$R_d = \left(\frac{\gamma^2 s_d s_e}{\tilde{s}_e \tilde{s}_c} + \frac{\gamma s_d s_e}{\tilde{s}_e s_c} + \frac{\gamma s_d}{\tilde{s}_e} + \frac{\gamma s_d}{\tilde{s}_c} + \frac{s_d}{s_c}\right) R_c.$$

Based on the condition

$$R_d + R_e + R_c = R,$$

that is,

$$\left(\frac{\gamma^2 s_d s_e}{\tilde{s}_e \tilde{s}_c} + \frac{\gamma s_d s_e}{\tilde{s}_e s_c} + \frac{\gamma s_d}{\tilde{s}_e} + \frac{\gamma s_d}{\tilde{s}_c} + \frac{s_d}{s_c} + \frac{\gamma s_e}{\tilde{s}_c} + \frac{s_e}{s_c} + 1\right) R_c = R,$$

we get

$$R_c = R\left(\frac{\gamma^2 s_d s_e}{\tilde{s}_e \tilde{s}_c} + \frac{\gamma s_d s_e}{\tilde{s}_e s_c} + \frac{\gamma s_d}{\tilde{s}_e} + \frac{\gamma(s_d + s_e)}{\tilde{s}_c} + \frac{s_d + s_e}{s_c} + 1\right)^{-1},$$

and

$$T_d = T_e = T_c = \frac{\frac{\gamma^2 s_e}{\tilde{s}_e \tilde{s}_c} + \frac{\gamma s_e}{\tilde{s}_e s_c} + \frac{\gamma}{\tilde{s}_e} + \frac{\gamma}{\tilde{s}_c} + \frac{1}{s_c}}{\frac{\gamma^2 s_d s_e}{\tilde{s}_e \tilde{s}_c} + \frac{\gamma s_d s_e}{\tilde{s}_e s_c} + \frac{\gamma s_d}{\tilde{s}_e} + \frac{\gamma(s_d + s_e)}{\tilde{s}_c} + \frac{s_d + s_e}{s_c} + 1} R.$$

The proof can be completed by straightforward algebraic manipulation. □

We would like to mention that the lower bound in the above theorem typically cannot be achieved by an optimal schedule, due to the discrete nature of tasks, i.e., R and D cannot be divided arbitrarily.

3.2.2 Heuristic Schedule

We consider a special case: $\gamma_i = \gamma$, $d_i = \gamma r_i$, for all $1 \le i \le n$.

Recall that τ_i is the execution time of task t_i. On the mobile device, it is

$$\tau_i = \frac{r_i}{s_d}.$$

On the edge server, it is

$$\tau_i = \frac{d_i}{\tilde{s}_e} + \frac{r_i}{s_e} = \frac{\gamma r_i}{\tilde{s}_e} + \frac{r_i}{s_e} = \left(\frac{\gamma}{\tilde{s}_e} + \frac{1}{s_e}\right) r_i = \frac{r_i}{\bar{s}_e},$$

where

$$\bar{s}_e = \frac{\tilde{s}_e s_e}{\gamma s_e + \tilde{s}_e}$$

is the *effective computation speed* of the edge server. On the cloud server, it is

$$\tau_i = \frac{d_i}{\tilde{s}_e} + \frac{d_i}{\tilde{s}_c} + \frac{r_i}{s_c} = \frac{\gamma r_i}{\tilde{s}_e} + \frac{\gamma r_i}{\tilde{s}_c} + \frac{r_i}{s_c} = \left(\frac{\gamma}{\tilde{s}_e} + \frac{\gamma}{\tilde{s}_c} + \frac{1}{s_c}\right) r_i = \frac{r_i}{\bar{s}_c},$$

where

$$\bar{s}_c = \frac{\tilde{s}_e \tilde{s}_c s_c}{\gamma \tilde{s}_c s_c + \gamma \tilde{s}_e s_c + \tilde{s}_e \tilde{s}_c} = \frac{\tilde{s}_e \tilde{s}_c s_c}{\gamma(\tilde{s}_e + \tilde{s}_c) s_c + \tilde{s}_e \tilde{s}_c}$$

is the effective computation speed of the cloud server. Therefore, we have

$$\tau_i \le \frac{r_i}{\min\{s_d, \bar{s}_e, \bar{s}_c\}}.$$

Let τ be the maximum execution time of any task anywhere, which is

$$\tau = \frac{r}{\min\{s_d, \bar{s}_e, \bar{s}_c\}},$$

where

$$r = \max\{r_1, r_2, \ldots, r_n\}.$$

We also notice that excluding the wireless communication time, the cloud server takes

$$\frac{d_i}{\tilde{s}_c} + \frac{r_i}{s_c} = \frac{\gamma r_i}{\tilde{s}_c} + \frac{r_i}{s_c} = \left(\frac{\gamma}{\tilde{s}_c} + \frac{1}{s_c}\right) r_i = \frac{r_i}{\bar{\bar{s}}_c}$$

time to process task t_i, where

$$\bar{\tilde{s}}_c = \frac{\tilde{s}_c s_c}{\gamma s_c + \tilde{s}_c}.$$

The following theorem gives an upper bound for the heuristic schedule length $T(S)$ of S.

Theorem 4. *If $d_i = \gamma r_i$, for all $1 \le i \le n$, the heuristic schedule length $T(S)$ has the following upper bound:*

$$T(S) \le BR + \tau,$$

where

$$B = \frac{\tilde{s}_e + \gamma(s_e + \bar{\tilde{s}}_c)}{s_d\tilde{s}_e + \gamma s_d(s_e + \bar{\tilde{s}}_c) + \tilde{s}_e(s_e + \bar{\tilde{s}}_c)}.$$

Proof. For convenience, let $T = T(S) = \max\{T_d, T_e, T_c\}$ be the schedule length of our heuristic algorithm. Assume that t_i is the last task, such that after t_i is added to S_d or S_e or S_c, T becomes $T(S)$. It is clear that the mobile device, the edge server, and the cloud server are all busy (i.e., not available) up to time $T - \tau_i$; otherwise, t_i could be added to another place and T can possibly be reduced. Since the mobile device is busy during $[0, T - \tau_i]$, i.e., $T_d \ge T - \tau_i$, we have

$$R_d = s_d T_d \ge s_d(T - \tau_i).$$

Let R'_e and R'_c be the workload on the edge server and the cloud server before t_i is added, and

$$T' = \frac{D'_e + D'_c}{\tilde{s}_e} = \frac{\gamma(R'_e + R'_c)}{\tilde{s}_e},$$

be the wireless communication time. Since the edge server and the cloud server are all busy in computation during $[T,' T - \tau_i]$, we have

$$R'_e + R'_c \ge (s_e + \bar{\tilde{s}}_c)(T - \tau_i - T'),$$

that is,

$$R'_e + R'_c \ge (s_e + \bar{\tilde{s}}_c)(T - \tau_i) - (s_e + \bar{\tilde{s}}_c)\frac{\gamma(R'_e + R'_c)}{\tilde{s}_e},$$

or,

$$(R'_e + R'_c)\left(1 + \frac{\gamma(s_e + \bar{\tilde{s}}_c)}{\tilde{s}_e}\right) \ge (s_e + \bar{\tilde{s}}_c)(T - \tau_i),$$

which gives

$$R'_e + R'_c \ge \frac{\tilde{s}_e(s_e + \bar{\tilde{s}}_c)}{\tilde{s}_e + \gamma(s_e + \bar{\tilde{s}}_c)}(T - \tau_i).$$

Consequently, we have

$$\begin{aligned} R &= R_d + R_e + R_c \ge R_d + R'_e + R'_c \\ &\ge \left(s_d + \frac{\tilde{s}_e(s_e + \bar{\tilde{s}}_c)}{\tilde{s}_e + \gamma(s_e + \bar{\tilde{s}}_c)}\right)(T - \tau_i), \end{aligned}$$

which implies that

$$T \le \left(\frac{\tilde{s}_e + \gamma(s_e + \bar{\tilde{s}}_c)}{s_d\tilde{s}_e + \gamma s_d(s_e + \bar{\tilde{s}}_c) + \tilde{s}_e(s_e + \bar{\tilde{s}}_c)}\right)R + \tau_i.$$

The proof can be completed by noticing that $\tau_i \le \tau$. □

It can be verified by straightforward algebraic manipulation that B in Theorem 4 is identical to that in Theorem 3.

3.2.3 Asymptotic Performance Bound

We say that B is an *absolute performance bound* if

$$\frac{T(S)}{T^*(S)} \le B,$$

for all S. We say that B is an *asymptotic performance bound* if

$$\lim_{R/r\to\infty} \frac{T(S)}{T^*(S)} \le B.$$

When S is a set of random tasks, if

$$\boldsymbol{E}\left[\frac{T(S)}{T^*(S)}\right] \le B,$$

B is an *average absolute performance bound*, and if

$$\lim_{R/r\to\infty} \boldsymbol{E}\left[\frac{T(S)}{T^*(S)}\right] \le B,$$

B is an *average asymptotic performance bound*.

Let

$$\gamma' = \min\{\gamma_1, \gamma_2, \ldots, \gamma_n\},$$

and

$$\gamma'' = \max\{\gamma_1, \gamma_2, \ldots, \gamma_n\}.$$

The interval $[\gamma,' \gamma'']$ indicates the level of communication heterogeneity. The wider the interval, the much the communication heterogeneity.

The lower/upper bound $B(\gamma)$ in Theorems 3 and 4 is treated as a function of γ. The following theorem gives an asymptotic performance bound for our heuristic algorithm for task scheduling with device-edge-cloud fusion.

Theorem 5. *Our heuristic algorithm for task scheduling with device-edge-cloud fusion has the following asymptotic performance bound:*

$$\lim_{R/r\to\infty} \frac{T(S)}{T^*(S)} \le \frac{B(\gamma'')}{B(\gamma')}.$$

Proof. Consider $S' = \{t'_1, t'_2, \ldots, t'_n\}$, where $t'_i = (d'_i, r_i)$, with $d'_i = \gamma' r_i$, for all $1 \le i \le n$. It is clear that a schedule for S is also applicable to S', with part of the communication time of t_i (i.e., $(t_i - t'_i)/\tilde{s}_e$ and $(t_i - t'_i)/\tilde{s}_c$) to be wasted. Furthermore, S' may have a shorter schedule. Therefore, the optimal schedule length of S' is no longer than the optimal schedule length of S, i.e.,

$$T^*(S') \le T^*(S).$$

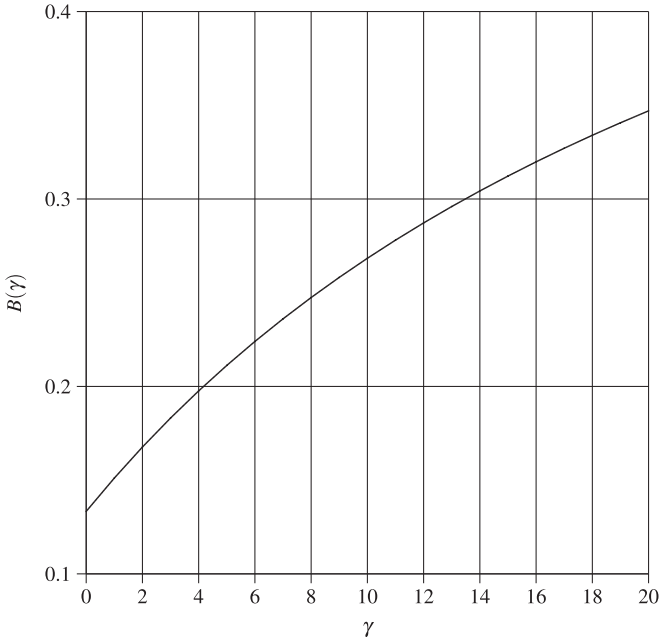

Fig. 3. $B(\gamma)$ as an increasing function of γ.

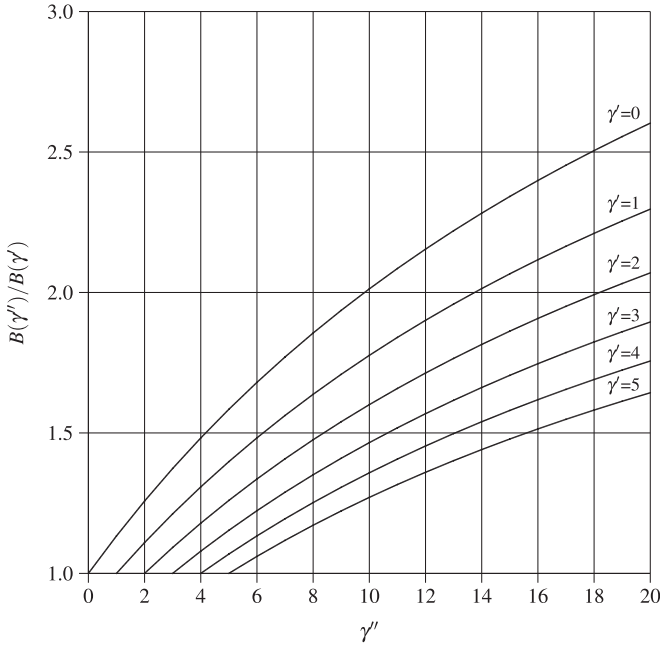

Fig. 4. $B(\gamma'')/B(\gamma')$ versus γ''.

By Theorem 3, the optimal schedule length of S' has lower bound $B(\gamma')R$, i.e.,

$$T^*(S') \geq B(\gamma')R.$$

Hence, we get

$$T^*(S) \geq B(\gamma')R.$$

Consider $S'' = \{t_1'', t_2'', \ldots, t_n''\}$, where $t_i'' = (d_i'', r_i)$, with $d_i'' = \gamma'' r_i$, for all $1 \leq i \leq n$. It is clear that a schedule for S'' is also applicable to S, with part of the communication time of t_i (i.e., $(t_i'' - t_i)/\tilde{s}_e$ and $(t_i'' - t_i)/\tilde{s}_c$) to be wasted. Furthermore, S may have a shorter schedule. Therefore, the heuristic schedule length of S is no longer than the heuristic schedule length of S'', i.e.,

$$T(S) \leq T(S'').$$

By Theorem 4, the heuristic schedule length of S'' has upper bound $B(\gamma'')R + \tau$, i.e.,

$$T(S'') \leq B(\gamma'')R + \tau.$$

Hence, we get

$$T(S) \leq B(\gamma'')R + \tau.$$

Therefore, we have

$$\frac{T(S)}{T^*(S)} \leq \frac{B(\gamma'')R + \tau}{B(\gamma')R} = \frac{B(\gamma'')}{B(\gamma')} + \frac{\tau}{B(\gamma')R},$$

and

$$\lim_{R/r \to \infty} \frac{T(S)}{T^*(S)} \leq \frac{B(\gamma'')}{B(\gamma')}.$$

The proof is completed. □

3.3 Performance Data

In this section, we present numerical data and simulation results.

3.3.1 Numerical Data

The lower/upper bound $B(\gamma)$ in Theorems 3 and 4 can be treated as a function of γ. Fig. 3 shows $B(\gamma)$ for $0 \leq \gamma \leq 20$, with the following parameters: $s_d = 1.5$ BI/second, $s_e = 2.5$ BI/second, $s_c = 3.5$ BI/second, $\tilde{s}_e = 40$ Mbps, $\tilde{s}_c = 90$ Mbps. It is clear that $B(\gamma)$ is an increasing function of γ. For $\gamma \in [0, \infty)$, we have

$$B(\gamma) \in \left[\frac{1}{s_d + s_e + s_c}, \frac{1}{s_d}\right).$$

When $\gamma = 0$, we have

$$B(\gamma) = \frac{1}{s_d + s_e + s_c} = 0.13333\ldots.$$

When $\gamma \to \infty$, we have

$$B(\gamma) \to \frac{1}{s_d} = 0.66666\ldots.$$

Fig. 4 shows the asymptotic performance bound $B(\gamma'')/B(\gamma')$, for $\gamma' = 0, 1, \ldots, 5$, and $\gamma' \leq \gamma'' \leq 20$. It is clear that $B(\gamma'')/B(\gamma')$ is an increasing function of γ''. However, it is a decreasing function of γ'. Actually, slight increment of γ' significantly reduces the asymptotic performance bound, thus strengthening its tightness.

3.3.2 Simulation Results

In this section, we demonstrate simulation results.

Our experiments also cover two other heuristic algorithms, which are more difficult to analyze, but whose performance can be easily evaluated experimentally.

Algorithm 2. It is clear that if t_i is the last task scheduled for execution, then t_i should be added to S_d or S_e or S_c, such

that the total execution time T is the minimum. Otherwise, t_i can be moved to another place, and the total execution time T can be reduced. This observation provides the basis of another heuristic algorithm, in which, line (4) in Algorithm 1 is changed to:

add t_i to S_d or S_e or S_c, such that

the new T_d or T_e or T_c (after t_i is added) is the minimum;

Notice that in Algorithm 1, line (4) is actually:

add t_i to S_d or S_e or S_c, such that

T_d or T_e or T_c (before t_i is added) is the minimum;

Algorithm 3. The set of tasks are sorted into a list $L = (t_1, t_2, \ldots, t_n)$, where

$$d_1/r_1 \geq d_2/r_2 \geq \cdots \geq d_n/r_n.$$

Tasks in the left end are communication-intensive, which are suitable for local execution. Tasks in the right end are computation-intensive, which are suitable for remote execution. Hence, when scheduling the next task, the mobile device picks the next task from the left end, the edge server and the cloud server pick the next task from the right end.

The time complexity of Algorithm 2 is $O(n)$.

The time complexity of Algorithm 3 is $O(n \log n)$ (mainly for sorting in line (3)).

Algorithm 2. Task Scheduling With Device-Edge-Cloud Fusion

Input: $S = \{t_1, t_2, \ldots, t_n\}$, $s_d, s_e, s_c, \tilde{s}_e, \tilde{s}_c$.
Output: A schedule (S_d, S_e, S_c), such that T is minimized.

$S_d \leftarrow \emptyset$, $S_e \leftarrow \emptyset$, $S_c \leftarrow \emptyset$; (1)
$T_d \leftarrow 0$, $T_e \leftarrow 0$, $T_c \leftarrow 0$; (2)
for $(i = 1; i \leq n; i{+}{+})$ **do** (3)
  $T'_d \leftarrow T_d + r_i/s_d$, $T'_e \leftarrow T_e + d_i/\tilde{s}_e + r_i/s_e$,
    $T'_c \leftarrow T_c + d_i/\tilde{s}_e + d_i/\tilde{s}_c + r_i/s_c$; (4)
  add t_i to S_d or S_e or S_c,
    where T'_d or T'_e or T'_c is the smallest; (5)
  update T_d or T_e or T_c; (6)
end do (7)

Algorithm 3. Task Scheduling With Device-Edge-Cloud Fusion

Input: $S = \{t_1, t_2, \ldots, t_n\}$, $s_d, s_e, s_c, \tilde{s}_e, \tilde{s}_c$.
Output: A schedule (S_d, S_e, S_c), such that T is minimized.

$S_d \leftarrow \emptyset$, $S_e \leftarrow \emptyset$, $S_c \leftarrow \emptyset$; (1)
$T_d \leftarrow 0$, $T_e \leftarrow 0$, $T_c \leftarrow 0$; (2)
make a list of tasks $L = (t_1, t_2, \ldots, t_n)$,
  such that $d_1/r_1 \geq d_2/r_2 \geq \cdots \geq d_n/r_n$; (3)
$j \leftarrow 1$, $k \leftarrow n$; (4)
for $(i = 1; i \leq n; i{+}{+})$ **do** (5)
  $T'_d \leftarrow T_d + r_j/s_d$, $T'_e \leftarrow T_e + d_k/\tilde{s}_e + r_k/s_e$,
    $T'_c \leftarrow T_c + d_k/\tilde{s}_e + d_k/\tilde{s}_c + r_k/s_c$; (6)
  add t_j to S_d, or add t_k to S_e or S_c,
    where T'_d or T'_e or T'_c is the smallest; (7)
  update T_d or T_e or T_c; (8)
  $j \leftarrow j + 1$ or $k \leftarrow k - 1$; (9)
end do (10)

The parameters are set as follows. The computation speeds are $s_d = 1.5$ BI/second, $s_e = 2.5$ BI/second, and $s_c = 3.5$ BI/second. The communication speeds are $\tilde{s}_e = 40$ Mbps and $\tilde{s}_c = 90$ Mbps. The computation power consumption model has $\xi = 0.1$, $\alpha = 2.0$, and $P_s = 50$ mW, which give computation power $P = \xi s_d^{\alpha} + P_s = 275$ mW. The communication power consumption model has $w = 30$ Mbps and $\beta = 2.0$ W^{-1}, which give transmission power $P_t = (2^{\tilde{s}_e/w} - 1)/\beta = 760$ mW. (These parameters are consistent with those reported in [3].)

TABLE 2
Simulation Results for Task Scheduling With D-E-C Fusion (Average Absolute Performance Bounds for Execution Time)

n	Algorithm 1	Algorithm 2	Algorithm 3
10	1.35430	1.32393	1.28648
20	1.28627	1.26547	1.22129
30	1.26054	1.24640	1.20399
40	1.24864	1.23433	1.19126
50	1.24284	1.23058	1.18487
60	1.23759	1.22519	1.18064
70	1.23307	1.22197	1.17715
80	1.22886	1.21915	1.17487
90	1.22795	1.21886	1.17300
100	1.22679	1.21741	1.17123

For each $n = 10, 20, \ldots, 100$, we generate $N = 500$ sets of independent and identically distributed random tasks. For each random task $t_i = (d_i, r_i)$, r_i is uniformly distributed in the range [1.0,4.0] GI, and $d_i = \gamma_i r_i$, where γ_i is uniformly distributed in the range $[\gamma, ' \gamma'']$, with $\gamma' = 1.0$ MB/GI and $\gamma'' = 5.0$ MB/GI.

In Table 2, we show our simulation results on execution time.

For each set S of tasks, we apply each of the three heuristic algorithms and record the ratio $T(S)/(BR)$, where BR is the lower bound for $T^*(S)$ in Theorem 3. The data in the table are average absolute performance bounds for the N sets of random tasks. (Note: Since $T(S)/T^*(S) \leq T(S)/(BR)$, we have $\boldsymbol{E}[T(S)/T^*(S)] \leq \boldsymbol{E}[T(S)/(BR)]$. Thus, $\boldsymbol{E}[T(S)/(BR)]$ is an average absolute performance bound.) The maximum 99% confidence interval is $\pm 0.61179\%$.

As a reference, the asymptotic performance bound in Theorem 5 (which is also an average asymptotic performance bound) is $B(\gamma'')/B(\gamma') = 1.39779$.

We have the following observations.

- The average absolute performance bounds are very close to one. This means that all our heuristic algorithms perform very well for task scheduling with device-edge-cloud fusion with performance very close to that of the optimal algorithm.
- The average absolute performance bounds of all algorithms decrease as n increases. This means that all our heuristic algorithms perform better for more tasks.
- The average absolute performance bounds are all less than the asymptotic performance bound, simply because the asymptotic performance bound is based on an over-estimation of $T(S)$ and an under-estimation of $T^*(S)$, and is not tight.
- Algorithm 2 consistently (statistically) produces shorter schedules than Algorithm 1. This means that slight foreseeing of the future can improve performance.
- Algorithm 3 consistently (statistically) produces shorter schedules than Algorithm 2. This means that

TABLE 3
Simulation Results for Task Scheduling With D-E-C Fusion (Energy Consumption in Joule)

n	Algorithm 1	Algorithm 2	Algorithm 3
10	2.34518	2.20987	2.05744
20	4.62112	4.46392	4.12028
30	6.92316	6.75506	6.21306
40	9.19916	8.99602	8.26242
50	11.48448	11.24304	10.32024
60	13.76803	13.50717	12.40014
70	16.06110	15.76958	14.43439
80	18.36395	18.04269	16.52746
90	20.70127	20.37276	18.66144
100	22.98020	22.62647	20.71192

it is effective to assign communication-intensive tasks to the mobile device to eliminate long communication times, and to assign computation-intensive to tasks to the edge server and the cloud server to reduce communication times.

In Table 3, we further show our simulation results on energy consumption, although it is not a main concern in this section. The method to generate Table 3 is identical to that of Table 2. For each set of tasks, we apply each of the three heuristic algorithms and record its energy consumption. The data in the table are average energy consumption for the N sets of random tasks. The maximum 99% confidence interval is $\pm 1.17464\%$. We have the following observations.

- Algorithm 2 consistently (statistically) consumes less energy than Algorithm 1. Therefore, Algorithm 2 is superior to Algorithm 1 in terms of both execution time and energy consumption.
- Algorithm 3 consistently (statistically) consumes less energy than Algorithm 2. Therefore, Algorithm 3 is superior to Algorithm 2 in terms of both execution time and energy consumption.

To summarize, Algorithm 3 is the best heuristic algorithm developed in this paper for task scheduling with device-edge-cloud fusion in terms of both execution time and energy consumption.

We would like to emphasize that our heuristic algorithms perform better for tasks with less communication heterogeneity. In Table 4, we re-do Table 2 with $\gamma' = 2.0$ MB/GI and $\gamma'' = 4.0$ MB/GI. The maximum 99% confidence interval is $\pm 0.57785\%$. It is clear that the performance of all algorithms are improved.

TABLE 4
Simulation Results for Task Scheduling With D-E-C Fusion (Average Absolute Performance Bounds for Execution Time, $\gamma' = 2.0$, $\gamma'' = 4.0$)

n	Algorithm 1	Algorithm 2	Algorithm 3
10	1.22693	1.20120	1.18213
20	1.15612	1.14488	1.12515
30	1.13397	1.12757	1.10550
40	1.12600	1.11845	1.09673
50	1.11811	1.11312	1.09032
60	1.11425	1.10917	1.08771
70	1.11131	1.10659	1.08405
80	1.10935	1.10440	1.08280
90	1.10642	1.10224	1.08021
100	1.10464	1.10134	1.07940

4 Energy-Constrained Task Scheduling

In this section, we consider the problem of energy-constrained task scheduling with device-edge-cloud fusion.

4.1 Algorithm

In this section, we develop a heuristic algorithm for energy-constrained task scheduling with device-edge-cloud fusion.

The challenges of the problem include two aspects, namely, task scheduling (i.e., to determine a schedule (S_d, S_e, S_c)) and power allocation (i.e., to determine speeds s_d and $\tilde{s}_e$).

The following theorem gives the optimal T and the optimal speed setting if the schedule is known.

Theorem 6. *For a given schedule (S_d, S_e, S_c), the optimal T satisfies*

$$\xi \frac{R_d^\alpha}{T^{\alpha-1}} + P_s T + \frac{2^{((D_e+D_c)/w)/(T-T'')} - 1}{\beta}(T - T'') = \hat{E}, \quad (1)$$

where

$$T'' = \max\left\{\frac{R_e}{s_e}, \frac{D_c}{\tilde{s}_c} + \frac{R_c}{s_c}\right\}.$$

Proof. For a given schedule (S_d, S_e, S_c), s_d and $\tilde{s}_e$ are determined in such a way that

$$T_d = \max\{T_e, T_c\} = T,$$

and that $E = \hat{E}$ (see Fig. 5).

From

$$\frac{R_d}{s_d} = T,$$

we get

$$s_d = \frac{R_d}{T}.$$

From

$$\frac{D_e + D_c}{\tilde{s}_e} = T - T'',$$

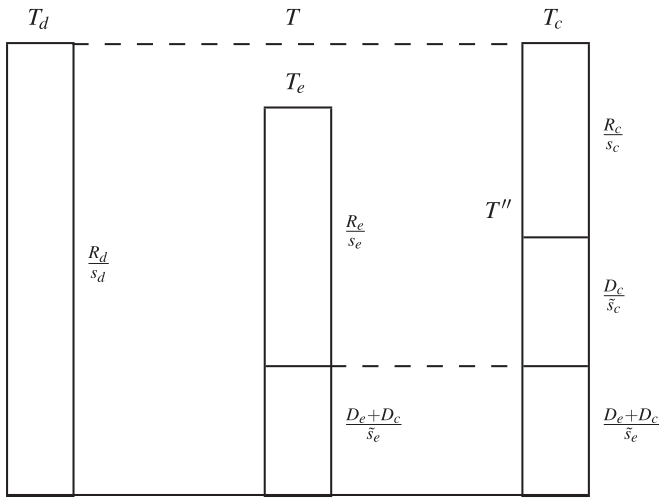

Fig. 5. $T_d = \max\{T_e, T_c\} = T$ with proper adjustment of s_d and $\tilde{s}_e$.

we get

$$\tilde{s}_e = \frac{D_e + D_c}{T - T''}.$$

Therefore, we have

$$E = \xi \frac{R_d^\alpha}{T^{\alpha-1}} + P_s T + \frac{2^{((D_e+D_c)/w)/(T-T'')} - 1}{\beta}(T - T'') = \hat{E}.$$

The theorem is proved. □

Notice that the above theorem also gives the optimal values of the speeds s_d and $\tilde{s}_e$.

Since the left-hand side of Eq. (1) is a decreasing function of T (i.e., longer time, less energy, which is less obvious analytically, but can be easily observed numerically), Eq. (1) can be solved using the bisection search method in $O(\log(\Delta/\epsilon))$ time, where Δ is the size of the initial search interval and ϵ is the accuracy requirement.

Let $T(S_d, S_e, S_c)$ obtained by Theorem 6 be a function of the schedule (S_d, S_e, S_c).

Theorem 6 provides the basis of our greedy algorithm. A unique feature of the algorithm is that the two subproblems (task scheduling and power allocation) are solved in an interleaved way. The determination of the speeds depends on the schedule, which in turn, depends on the optimal speed setting.

Algorithm 4. Energy-Constrained Task Scheduling With Device-Edge-Cloud Fusion

Input: $S = \{t_1, t_2, \ldots, t_n\}$, $s_e, s_c, \tilde{s}_c$, and $\hat{E}$.
Output: A schedule (S_d, S_e, S_c) and speeds $s_d, \tilde{s}_e$, such that T is minimized and that $E = \hat{E}$.

$S_d \leftarrow \emptyset$, $S_e \leftarrow \emptyset$, $S_c \leftarrow \emptyset$; (1)
make a list of tasks $L = (t_1, t_2, \ldots, t_n)$,
  such that $d_1/r_1 \geq d_2/r_2 \geq \cdots \geq d_n/r_n$; (2)
for $(i = 1; i \leq n; i$++$)$ **do** (3)
  calculate $T(S_d \cup \{t_i\}, S_e, S_c)$, $T(S_d, S_e \cup \{t_i\}, S_c)$,
    $T(S_d, S_e, S_c \cup \{t_i\})$ by solving Eq. (1); (4)
  add t_i to S_d or S_e or S_c, such that T is the minimum; (5)
end do; (6)
calculate $T, s_d, \tilde{s}_e$. (7)

Algorithm 4 presents our greedy algorithm to solve the problem of energy-constrained task scheduling with device-edge-cloud fusion.

Initially, S_d, S_e, S_c are set to be empty sets. The set of tasks are ordered into a list $L = (t_1, t_2, \ldots, t_n)$, such that $d_1/r_1 \geq d_2/r_2 \geq \cdots \geq d_n/r_n$. For each task t_i, $T(S_d \cup \{t_i\}, S_e, S_c)$, $T(S_d, S_e \cup \{t_i\}, S_c)$, and $T(S_d, S_e, S_c \cup \{t_i\})$ are calculated, and t_i is added to one of S_d, S_e, S_c, such that T (obtained from Theorem 6) is the minimum.

The time complexity of Algorithm 4 is $O(n\log n + n\log(\Delta/\epsilon))$.

4.2 Analysis

Let $B(s_d, \tilde{s}_e, \gamma)$ denote the lower/upper bound in Theorems 3 and 4, which is treated as a function of $s_d, \tilde{s}_e, \gamma$.

The following theorem gives a lower bound for the optimal schedule length T^*.

Theorem 7. *The optimal schedule length T^* has the following lower bound*

$$T^* \geq B^* R,$$

where B^* is the minimum value of $B(s_d, \tilde{s}_e, \gamma')$ subject to the constraint

$$F(s_d, \tilde{s}_e) = (\xi s_d^\alpha + P_s) B(s_d, \tilde{s}_e, \gamma') + \gamma' \left(\frac{2^{\tilde{s}_e/w} - 1}{\beta \tilde{s}_e}\right)(1 - s_d B(s_d, \tilde{s}_e, \gamma')) = \frac{\hat{E}}{R}. \quad (2)$$

Proof. We have already known from the proof of Theorem 5 that for given s_d and $\tilde{s}_e$, the optimal schedule length $T^*(S)$ has the following lower bound:

$$T^*(S) \geq T^*(S') \geq B(s_d, \tilde{s}_e, \gamma') R.$$

As a matter of fact, the above inequality is still valid for energy-constrained task scheduling with device-edge-cloud fusion. The reason is that any solution for S (i.e., a schedule (S_d, S_e, S_c) plus speeds $s_d, \tilde{s}_e$) is also applicable to S'. Actually, the total energy consumption of S' may be less than $\hat{E}$, since some communication time is not utilized. Therefore, with the same energy constraint $\hat{E}$, $T^*(S')$ could be shorter than $T^*(S)$.

When s_d and $\tilde{s}_e$ are variables, we need to find s_d and $\tilde{s}_e$ which minimize $B(s_d, \tilde{s}_e, \gamma')$, subject to the constraint

$$E(s_d, \tilde{s}_e) = \left(\frac{\xi s_d^\alpha + P_s}{s_d}\right) R_d + \left(\frac{2^{\tilde{s}_e/w} - 1}{\beta \tilde{s}_e}\right)(D_e + D_c) = \hat{E}.$$

From the proof of Theorem 3, we know that

$$R_d = s_d B(s_d, \tilde{s}_e, \gamma') R,$$

and

$$D_e + D_c = \gamma'(R_r + R_c) = \gamma'(R - R_d) = \gamma'(1 - s_d B(s_d, \tilde{s}_e, \gamma'))R.$$

Therefore, we get

$$E(s_d, \tilde{s}_e) = (\xi s_d^\alpha + P_s) B(s_d, \tilde{s}_e, \gamma') R + \left(\frac{2^{\tilde{s}_e/w} - 1}{\beta \tilde{s}_e}\right)\gamma'(1 - s_d B(s_d, \tilde{s}_e, \gamma'))R = \hat{E},$$

which is equivalent to the constraint in the theorem. This proves the theorem. □

No analytical form of B^* is available. However, B^* can be obtained numerically. To this end, we need

$$\frac{\partial B(s_d, \tilde{s}_e, \gamma')}{\partial s_d} = \phi \frac{\partial F(s_d, \tilde{s}_e)}{\partial s_d}, \quad (3)$$

and

$$\frac{\partial B(s_d, \tilde{s}_e, \gamma')}{\partial \tilde{s}_e} = \phi \frac{\partial F(s_d, \tilde{s}_e)}{\partial \tilde{s}_e}, \quad (4)$$

where ϕ is a Lagrange multiplier.

In the following, we calculate the four derivatives. First, we notice that

$$\frac{\partial B(s_d, \tilde{s}_e, \gamma')}{\partial s_d} = -B^2(s_d, \tilde{s}_e, \gamma').$$

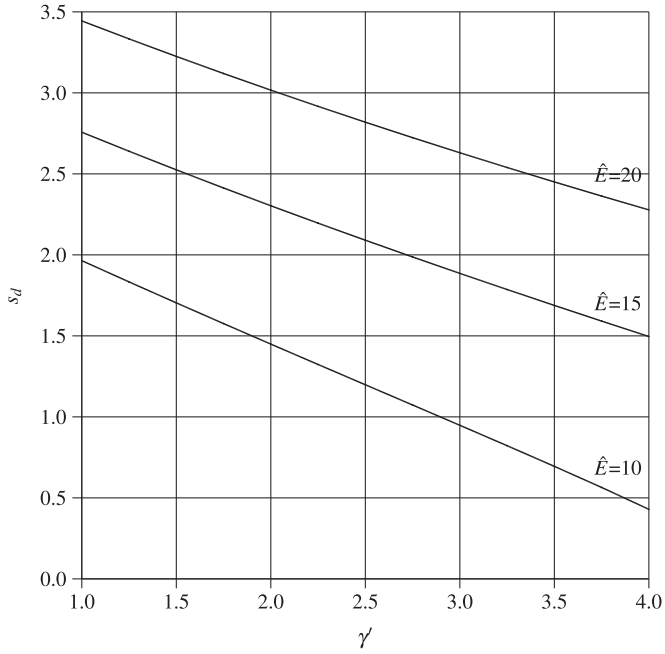

Fig. 6. s_d versus γ'.

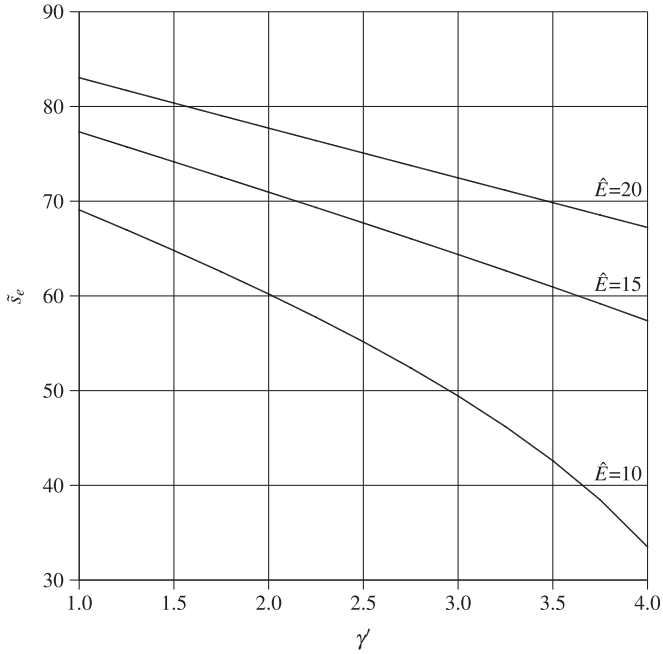

Fig. 7. $\tilde{s}_e$ versus γ'.

Second, we have

$$\frac{\partial F(s_d,\tilde{s}_e)}{\partial s_d} = \xi\alpha s_d^{\alpha-1} B(s_d,\tilde{s}_e,\gamma') - (\xi s_d^{\alpha}+P_s)B^2(s_d,\tilde{s}_e,\gamma') + \gamma'\left(\frac{2^{\tilde{s}_e/w}-1}{\beta\tilde{s}_e}\right)s_d B^2(s_d,\tilde{s}_e,\gamma').$$

Third, for convenience, we rewrite $B(s_d,\tilde{s}_e,\gamma')$ as

$$B(s_d,\tilde{s}_e,\gamma') = \frac{Q_1\tilde{s}_e+Q_2}{Q_3\tilde{s}_e+Q_4},$$

where

$$\begin{aligned} Q_1 &= \gamma' s_c + \tilde{s}_c,\\ Q_2 &= \gamma'^2 s_e s_c + \gamma' s_e\tilde{s}_c + \gamma'\tilde{s}_c s_c,\\ Q_3 &= \gamma'(s_d+s_e)s_c + (s_d+s_e)\tilde{s}_c + \tilde{s}_c s_c,\\ Q_4 &= \gamma'^2 s_d s_e s_c + \gamma' s_d s_e\tilde{s}_c + \gamma' s_d\tilde{s}_c s_c. \end{aligned}$$

Then, we have

$$\frac{\partial B(s_d,\tilde{s}_e,\gamma')}{\partial\tilde{s}_e} = \frac{Q_1}{Q_3\tilde{s}_e+Q_4} - \frac{Q_3(Q_1\tilde{s}_e+Q_2)}{(Q_3\tilde{s}_e+Q_4)^2}.$$

Finally, we have

$$\frac{\partial F(s_d,\tilde{s}_e)}{\partial\tilde{s}_e} = (\xi s_d^{\alpha}+P_s)\frac{\partial B(s_d,\tilde{s}_e,\gamma')}{\partial\tilde{s}_e} + \frac{\gamma'}{\beta}\left(\frac{2^{\tilde{s}_e/w}\ln 2}{w\tilde{s}_e} - \frac{2^{\tilde{s}_e/w}-1}{\tilde{s}_e^2}\right)(1-s_d B(s_d,\tilde{s}_e,\gamma')) - \gamma'\left(\frac{2^{\tilde{s}_e/w}-1}{\beta\tilde{s}_e}\right)s_d\frac{\partial B(s_d,\tilde{s}_e,\gamma')}{\partial\tilde{s}_e}.$$

We now have a non-linear system of three equations (i.e., Eqs. (2), (3), (4)) with three unknowns (i.e., s_d, $\tilde{s}_e$, and ϕ). We develop the following unique method to solve the equations. Let

$$\phi_1 = \frac{\partial B(s_d,\tilde{s}_e,\gamma')}{\partial s_d}\Big/\frac{\partial F(s_d,\tilde{s}_e)}{\partial s_d},$$

and

$$\phi_2 = \frac{\partial B(s_d,\tilde{s}_e,\gamma')}{\partial\tilde{s}_e}\Big/\frac{\partial F(s_d,\tilde{s}_e)}{\partial\tilde{s}_e}.$$

Given s_d, we can calculate $\tilde{s}_e$ using the condition $F(s_d,\tilde{s}_e) = \hat{E}/R$. For given s_d and $\tilde{s}_e$, we can calculate ϕ_1 and ϕ_2. It is observed that $\phi_1-\phi_2$ is an increasing function of s_d. Thus, we can find s_d using bisection search, such that $\phi_1-\phi_2=0$.

4.3 Performance Data

In this section, we present numerical data and simulation results.

4.3.1 Numerical Data

We set $R = 125$ GI, which is equivalent to $n = 50$ tasks.

For $\hat{E} = 10, 15, 20$ Joule, Figs. 6, 7, and 8 show s_d, $\tilde{s}_e$, and B^*R, where s_d and $\tilde{s}_e$ are the speeds which give B^*, i.e., the minimum value of $B(s_d,\tilde{s}_e,\gamma')$.

We have the following observations.

- For given energy constraint $\hat{E}$, as γ' increases, both s_d and $\tilde{s}_e$ decrease to accommodate more communication within the same energy budget. For given γ', as $\hat{E}$ increases, both s_d and $\tilde{s}_e$ increase, which reduce the execution time.
- For given energy constraint $\hat{E}$, as γ' increases, the lower bound B^*R increases due to longer communication time. For given γ', as $\hat{E}$ increases, the lower bound B^*R decreases due to increased computation and communication speeds.

4.3.2 Simulation Results

We would like to mention that any algorithm A for task scheduling with device-edge-cloud fusion can be easily

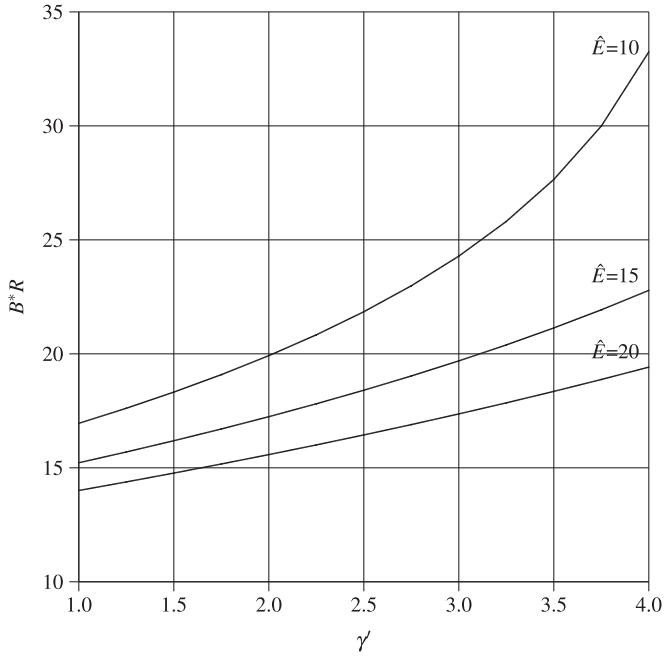

Fig. 8. B^*R versus γ'.

adapted for energy-constrained task scheduling with device-edge-cloud fusion. The adapted algorithm A' has two stages. In the first stage, we run algorithm A to determine a schedule (S_d, S_e, S_c) with some reasonable setting of s_d and $\tilde{s}_e$. In the second stage, we apply Theorem 6 to determine speeds s_d and $\tilde{s}_e$. For Algorithms 1 and 2, the time complexity of the adapted algorithm is $O(n + \log(\Delta/\epsilon))$. For Algorithm 3, the time complexity of the adapted algorithm is $O(n\log n + \log(\Delta/\epsilon))$.

In Table 5, we show our simulation results to compare the performance of Algorithms 1′–3′ and 4. We use the same parameter setting as that in Section 3.3.2.

For each $n = 10, 20, \ldots, 100$, we generate $N = 20000$ sets of random tasks. The energy constraint is $\hat{E} = 0.2n$ Joule. For each set S of tasks, we apply each of the four heuristic algorithms (the speed setting for Algorithms 1–3 is: $s_d = 1.5$ BI/second, $\tilde{s}_e = 40$ Mbps), record its execution time $T(S)$, calculate the lower bound B^*R for the optimal solution $T^*(S)$ using Theorem 7, and obtain the ratio $T(S)/(B^*R)$. The data in the table are average absolute performance bounds for the N sets of random tasks. The maximum 99% confidence interval is $\pm 0.56688\%$.

TABLE 5
Simulation Results for Energy-Constrained Task Scheduling With D-E-C Fusion (Average Absolute Performance Bounds for Execution Time)

n	Algorithm 1′	Algorithm 2′	Algorithm 3′	Algorithm 4
10	1.93314	1.69734	1.75338	1.43470
20	1.74684	1.61419	1.66368	1.39702
30	1.68515	1.58381	1.62414	1.37909
40	1.65535	1.57095	1.60309	1.37039
50	1.63721	1.56460	1.59067	1.36558
60	1.62707	1.56020	1.58193	1.36206
70	1.62174	1.55928	1.57938	1.36007
80	1.61602	1.55672	1.57417	1.35823
90	1.61219	1.55434	1.57074	1.35681
100	1.60864	1.55308	1.56820	1.35572

TABLE 6
Simulation Results for Energy-Constrained Task Scheduling With D-E-C Fusion (Average Absolute Performance Bounds for Execution Time, $\gamma' = 2.0$, $\gamma'' = 4.0$)

n	Algorithm 1′	Algorithm 2′	Algorithm 3′	Algorithm 4
10	1.61547	1.43474	1.48531	1.24878
20	1.46256	1.37350	1.41289	1.21926
30	1.41618	1.35814	1.38592	1.20642
40	1.39652	1.35156	1.37396	1.20067
50	1.38599	1.34765	1.36722	1.19684
60	1.37866	1.34582	1.36044	1.19456
70	1.37354	1.34428	1.35681	1.19265
80	1.37127	1.34423	1.35547	1.19166
90	1.36826	1.34379	1.35364	1.19078
100	1.36659	1.34366	1.35290	1.19009

We have the following observations.

- The average absolute performance bounds are reasonably close to one. This means that all our heuristic algorithms perform well for energy-constrained task scheduling with device-edge-cloud fusion with performance close to that of the optimal algorithm.
- The average absolute performance bounds of all algorithms decrease as n increases. This means that all our heuristic algorithms perform better for more tasks.
- The average absolute performance bounds of Algorithms 1′–3′ are higher than those in Table 2. This means that there is room for improving the performance of Algorithms 1′–3′, and tightening our lower bound.
- Algorithm 4 performs noticeably better than Algorithms 1′–3′, due to its interleaved strategy to determine a schedule and the computation/communication speeds.
- Algorithm 3′ is inferior to Algorithm 2′. This means that better performance for fixed speeds does not guarantee superiority when the speeds are variable.

In Table 6, we re-do Table 5 for tasks with with less communication heterogeneity with $\gamma' = 2.0$ MB/GI and $\gamma'' = 4.0$ MB/GI. The maximum 99% confidence interval is $\pm 0.49142\%$. It is clear that the performance of all algorithms are noticeably improved.

5 Conclusion

We have studied the NP-hard problems of task scheduling with device-edge-cloud fusion (without energy concern) and energy-constrained task scheduling with device-edge-cloud fusion as combinatorial optimization problems. We have developed heuristic algorithms for both problems. One strong and unique feature of our investigation is to derive lower bounds for optimal solutions and to compare heuristic solutions with optimal solutions. We found that the performance of our heuristic algorithms are very close to that of optimal algorithms.

There are several interesting and important directions for future work. First, we can consider multiple edge servers and multiple cloud servers in a mobile edge computing

environment. Second, we can research precedence constrained tasks, which makes task scheduling more complicated and challenging. Third, we can study multiple mobile devices, which compete for computation and communication resources. Fourth, we can design algorithms that allow overlapping of computation and communication (e.g., after an edge server receives a task, it starts to execute the task, and at the same time, receives the next task).

Acknowledgments

The comments and suggestions from the anonymous reviewers are acknowledged.

References

[1] F. Bu, Q. Zhang, L. T. Yang, and H. Yu, "An edge-cloud-aided high-order possibilistic c-means algorithm for Big Data clustering," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 12, pp. 3100–3109, Dec. 2020.

[2] Z. Cao, H. Zhang, B. Liu, and B. Sheng, "Revenue sharing in edge-cloud systems: A game-theoretic perspective," *Comput. Netw.*, vol. 176, 2020, Art. no. 107286.

[3] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Conf. Annu. Tech. Conf.*, 2010, Art. no. 21.

[4] L. Chunlin, H. Sun, C. Yi, and Y. Luo, "Edge cloud resource expansion and shrinkage based on workload for minimizing the cost," *Future Gener. Comput. Syst.*, vol. 101, pp. 327–340, 2019.

[5] C. Ding, A. Zhou, Y. Liu, R. N. Chang, C.-H. Hsu, and S. Wang, "A cloud-edge collaboration framework for cognitive service," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1489–1499, Third Quarter 2022.

[6] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[7] M. Du, Y. Wang, K. Ye, and C. Xu, "Algorithmics of cost-driven computation offloading in the edge-cloud environment," *IEEE Trans. Comput.*, vol. 69, no. 10, pp. 1519–1532, Oct. 2020.

[8] R. Du, Y. Liu, L. Liu, and W. Du, "A lightweight heterogeneous network clustering algorithm based on edge computing for 5G," *Wireless Netw.*, vol. 26 pp. 1631–1641, 2020.

[9] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Comput. Surv.*, vol. 52, no. 5, 2019, Art. no. 94.

[10] R. Fantacci and B. Picano, "A matching game with discard policy for virtual machines placement in hybrid cloud-edge architecture for industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7046–7055, Nov. 2020.

[11] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2687–2700, Nov. 2022.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability–A. Guide to the Theory of NP-Completeness*, New York, NY, USA: Freeman, 1979.

[13] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems," *IEEE Access*, vol. 7, pp. 78685–78697, 2019.

[14] Y. Hao, Y. Jiang, T. Chen, D. Cao, and M. Chen, "iTaskOffloading: Intelligent task offloading for a cloud-edge collaborative system," *IEEE Netw.*, vol. 33, no. 5, pp. 82–88, Sep./Oct. 2019.

[15] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud-MEC collaborative task offloading scheme with service orchestration," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5792–5805, Jul. 2020.

[16] Y. Huang, F. Wang, F. Wang, and J. Liu, "DeePar: A hybrid device-edge-cloud execution framework for mobile deep learning applications," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 892–897.

[17] C. Li, H. Sun, H. Tang, and Y. Luo, "Adaptive resource allocation based on the billing granularity in edge-cloud architecture," *Comput. Commun.*, vol. 145, pp. 29–42, 2019.

[18] K. Li, "Heuristic computation offloading algorithms for mobile users in fog computing," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 2, pp. 1–28, 2021.

[19] K. Li, "Scheduling precedence constrained tasks for mobile applications in fog computing," *IEEE Trans. Serv. Comput.*, early access, Jul. 19, 2022. doi: 10.1109/TSC.2022.3192095.

[20] F. P.-C. Lin and Z. Tsai, "Hierarchical edge-cloud SDN controller system with optimal adaptive resource allocation for load-balancing," *IEEE Syst. J.*, vol. 14, no. 1, pp. 265–276, Mar. 2020.

[21] B. Liu, W. Zhang, W. Chen, H. Huang, and S. Guo, "Online computation offloading and traffic routing for UAV swarms in edge-cloud computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8777–8791, Aug. 2020.

[22] R. Mahmud, S.N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated fog-cloud computing environments," *J. Parallel Distrib. Comput.*, vol. 135, pp. 177–190, 2020.

[23] K. Peng, H. Huang, S. Wan, and V. C. M. Leung, "End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment," *Wireless Netw.*, pp. 1–12, 2020.

[24] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[25] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, 2019, Art. no. 125.

[26] F. Song, M. Zhu, Y. Zhou, I. You, and H. Zhang, "Smart collaborative tracking for ubiquitous power IoT in edge-cloud interplay domain," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6046–6055, Jul. 2020.

[27] C. Sun et al., "Task offloading for end-edge-cloud orchestrated computing in mobile networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2020, pp. 1–6.

[28] S. Wang, S. Yang, and C. Zhao, "SurveilEdge: Real-time video query based on collaborative cloud-edge deep learning," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2519–2528.

[29] T. Wang, Y. Lu, J. Wang, H.-N. Dai, X. Zheng, and W. Jia, "EIHDP: Edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for IoT systems," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1285–1298, Aug. 2021.

[30] A. Younis, B. Qiu, and D. Pompili, "Latency-aware hybrid edge cloud framework for mobile augmented reality applications," *Proc. IEEE 17th Int. Conf. Sens., Commun., Netw.*, 2020, pp. 1–9.

[31] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.

Keqin Li (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York. He is also a national distinguished professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, Big Data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or coauthored more than 870 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds more than 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*. He is an AAIA fellow. He is also a member of Academia Europaea (Academician of the Academy of Europe).

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.