Journal of Grid Computing (2024) 22:60
https://doi.org/10.1007/s10723-024-09775-1

RESEARCH

Energy-Constrained DAG Scheduling on Edge and Cloud Servers with Overlapped Communication and Computation

Keqin Li

Received: 6 April 2024 / Accepted: 16 June 2024

Abstract Mobile edge computing (MEC) has been widely applied to numerous areas and aspects of human life and modern society. Many such applications can be represented as directed acyclic graphs (DAG). Device-edge-cloud fusion provides a new kind of heterogeneous, distributed, and collaborative computing environment to support various MEC applications. DAG scheduling is a procedure employed to effectively and efficiently manage and monitor the execution of tasks that have precedence constraints on each other. In this paper, we investigate the NP-hard problems of DAG scheduling and energy-constrained DAG scheduling on mobile devices, edge servers, and cloud servers by designing and evaluating new heuristic algorithms. Our contributions to DAG scheduling can be summarized as follows. First, our heuristic algorithms guarantee that all task dependencies are correctly followed by keeping track of the number of remaining predecessors that are still not completed. Second, our heuristic algorithms ensure that all wireless transmissions between a mobile device and edge/cloud servers are performed one after another. Third, our heuristic algorithms allow an edge/cloud server to start the execution of a task as soon as the transmission of the task is finished. Fourth, we derive a lower bound for the optimal makespan such that the solutions of our heuristic algorithms can be compared with optimal solutions. Our contributions to energy-constrained DAG scheduling can be summarized as follows. First, our heuristic algorithms ensure that the overall computation energy consumption and communication energy consumption does not exceed the given energy constraint. Second, our algorithms adopt an iterative and progressive procedure to determine appropriate computation speed and wireless communication speeds while generating a DAG schedule and satisfying the energy constraint. Third, we derive a lower bound for the optimal makespan and evaluate the performance of our heuristic algorithms in such a way that their heuristic solutions are compared with optimal solutions. To the author's knowledge, this is the first paper that considers DAG scheduling and energy-constrained DAG scheduling on edge and cloud servers with sequential wireless communications and overlapped communication and computation to minimize makespan.

Keywords Device-edge-cloud fusion · Directed acyclic graphs · Energy constraint · Heuristic algorithm · Makespan · Power consumption · Task scheduling

K. Li (✉)
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
e-mail: lik@newpaltz.edu

1 Introduction

1.1 Background and Challenges

Mobile edge computing (MEC) has been widely applied to numerous areas and aspects of human life and modern society, such as augmented reality, autonomous vehicles, healthcare monitoring, industrial IoT, smart homes, smart cities, and video surveillance. Many such applications can be represented as *directed acyclic graphs* (DAG) [2,4,11,12,15,18–20,22,27]. Typically, an MEC application includes different types of tasks with interdependencies, e.g., data collection, data fusion, data preprocessing, decision-making, event recognition, monitoring and sensing, object detection, object recognition, predictive maintenance, and process control.

Device-edge-cloud fusion provides a new kind of heterogeneous, distributed, and collaborative computing environment to support various MEC applications that include and involve communication-intensive tasks, computation-intensive tasks, and data-intensive tasks [3,16,17,21,25,26]. A mobile device can handle real-time monitoring and initial data processing tasks for real-time control and response, immediate decision-making, and time reduction between data acquisition and analysis. Edge server processing can enhance responsiveness, minimize latency, and reduce the need for transmitting large volumes of data to the cloud. Non-real-time tasks and complex analytics can be offloaded to a cloud server for more extensive and comprehensive analysis and massive data storage.

DAG scheduling is a procedure employed to effectively and efficiently manage and monitor the execution of tasks that have precedence constraints on each other [1,9]. In the context of a device-edge-cloud collaborative computing platform, where computing tasks are distributed across *user equipments* (UE), *edge servers* (ES), and *cloud servers* (CS), DAG scheduling becomes critical and crucial for optimizing resource utilization in a device-edge-cloud collaborative computing platform and minimizing execution time in processing an MEC application with interdependent tasks.

There are several challenges for DAG scheduling on multiple heterogeneous edge and cloud servers. First, the precedence constraints among the tasks should be properly handled in the sense that a task can be scheduled only when all its predecessors are completed. Second, since tasks are all generated on a mobile device (i.e., a UE), task allocation and assignment to the edge and cloud servers can be accomplished only by sequential transmission from the mobile device to the edge and cloud servers via wireless communication. Third, to maximize the utilization of computation and communication resources, on the same server, wireless communication of one task can overlap with wired communication and computation of another task. Fourth, as in traditional scheduling theory, the optimization objective is the makespan and the performance of a heuristic algorithm should be compared with that of an optimal algorithm.

Energy-constrained DAG scheduling on multiple heterogeneous edge and cloud servers incurs additional difficulties and challenges. First, the total energy consumption to process and execute a DAG, which includes both computation energy consumption and communication energy consumption, cannot exceed a certain given energy budget. Second, the computation speed for a task executed locally on a UE or the wireless communication speed for a task executed remotely on an edge server or a cloud server needs to be decided together with a schedule. Third, as mentioned above, the makespan of a heuristic algorithm should be compared with that of an optimal algorithm, which is a major challenge for energy-constrained DAG scheduling with sequential wireless communications and overlapped communication and computation.

1.2 New Contributions

In this paper, we investigate the NP-hard problems of DAG scheduling and energy-constrained DAG scheduling on mobile devices, edge servers, and cloud servers by designing and evaluating new heuristic algorithms. Our contributions to DAG scheduling can be summarized as follows.

- First, our heuristic algorithms guarantee that all task dependencies are correctly followed by keeping track of the number of remaining predecessors that are still not completed.
- Second, our heuristic algorithms ensure that all wireless transmissions between a mobile device and edge/cloud servers are performed one after another.
- Third, our heuristic algorithms allow an edge/cloud server to start the execution of a task as soon as the transmission of the task is finished.
- Fourth, we derive a lower bound for the optimal makespan such that the solutions of our heuristic algorithms can be compared with optimal solutions.

Our contributions to energy-constrained DAG scheduling can be summarized as follows.

- First, our heuristic algorithms ensure that the overall computation energy consumption and communication energy consumption does not exceed the given energy constraint.
- Second, our algorithms adopt an iterative and progressive procedure to determine appropriate computation speed and wireless communication speeds while generating a DAG schedule and satisfying the energy constraint.
- Third, we derive a lower bound for the optimal makespan and evaluate the performance of our heuristic algorithms in such a way that their heuristic solutions are compared with optimal solutions.

To the author's knowledge, this is the first paper that considers DAG scheduling and energy-constrained DAG scheduling on edge and cloud servers with sequential wireless communications and overlapped communication and computation to minimize makespan. The primary purpose of this paper is to compare our heuristic schedules with optimal schedules, not to compare the heuristic schedules among themselves.

The rest of the paper is organized as follows. In Section 2, we describe our DAG scheduling models on edge and cloud servers, including the server model, the task model, and the communication and computation model. In Section 3, we define our DAG-scheduling problem, develop our heuristic algorithms, derive a lower bound for the optimal schedule length, and experimentally evaluate the performance of our heuristic algorithms. In Section 4, we describe the power consumption models and discuss the energy consumption and energy efficiency of our heuristic algorithms. In Section 5, we define our energy-constrained DAG scheduling problem, present our heuristic algorithms, derive a lower bound for the optimal schedule length, and conduct experimental performance evaluation. In Section 6, we review related research. In Section 7, we summarize the paper and point out some future research directions.

2 Scheduling Models

In this section, we describe our DAG scheduling models on edge and cloud servers, including the server model, the task model, and the communication and computation model. We also discuss the extensibility of our models. The appendix gives a summary of all notations and their definitions.

2.1 Server Model

A heterogeneous and distributed device-edge-cloud collaborative computing system has $m+1$ servers: $S_0, S_1, S_2, ..., S_m$. Figure 1 illustrates such a system, where we assume that there are m_1 edge servers (ES): $S_1, ..., S_{m_1}$, and m_2 cloud servers (CS): $S_{m_1+1}, ..., S_{m_1+m_2}$, with $m = m_1 + m_2$.

S_0 is the UE. S_j can be either an ES or a CS, where $1 \le j \le m$. s_j is the computation speed (measured by billion instructions per second (Bips)) of S_j, where $0 \le j \le m$. c_j is the wireless communication speed (measured by million bits per second (Mbps)) of S_j, where $1 \le j \le m$. w_j is the wired (i.e., the Internet) communication speed (measured by million bits per second (Mbps)) of S_j (if S_j is a CS). Each CS has a *communication frontend* (CF) to handle wireless communication.

For convenience, if S_j is an ES, S_j is also called ES_j; and if S_j is a CS, S_j is also called CS_j with its CF_j (see Fig. 1, where we have UE, ES_1, ..., ES_{m_1}, CS_{m_1+1}, ..., $\mathrm{CS}_{m_1+m_2}$, CF_{m_1+1}, ..., $\mathrm{CF}_{m_1+m_2}$).

2.2 Task Model

A directed acyclic graph (DAG) is represented as $G = (\mathcal{T}, \prec)$, where $\mathcal{T} = \{T_1, T_2, ..., T_n\}$ is a set of tasks and $\prec \subseteq \mathcal{T} \times \mathcal{T}$ is a set of precedence constraints. A task $T_i = (d_i, r_i)$, where $1 \le i \le n$, is specified by the amount of communication d_i (measured by million bits (MB)) and the amount of computation r_i (measured by billion instructions (BI)).

G is initially on the UE, i.e., all tasks are generated on a mobile device. A task can be executed on the UE or offloaded to an ES or a CS via wireless and wired communication for execution.

Scheduling independent tasks in a device-edge-cloud collaborative computing system has been considered in [16,17]. Scheduling precedence-constrained tasks is more difficult than scheduling independent tasks.

2.3 Communication and Computation Model

There are three task execution modes.

- Device execution – If T_i is executed on the UE, the execution time is $t_i = r_i / s_0$.

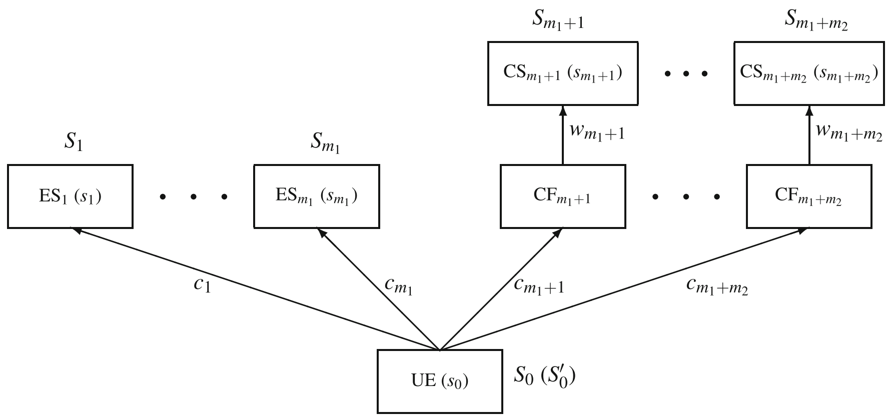


Fig. 1 A heterogeneous and distributed device-edge-cloud collaborative computing system

- Edge execution – If T_i is executed on ES_j, the execution time is $t_i = d_i/c_j + r_i/s_j$, where d_i/c_j is the wireless communication time and r_i/s_j is the computation time.
- Cloud execution – If T_i is executed on CS_j, the execution time is $t_i = d_i/c_j + d_i/w_j + r_i/s_j$, where d_i/c_j is the wireless communication time, d_i/w_j is the wired communication time, and r_i/s_j is the computation time.

Notice that all wireless communications are sequential, i.e., when tasks are offloaded from the UE to the ES and CS, the UE can only communicate with at most one S_j at a time, where $1 \le j \le m$. This is different from and more difficult than [13–15], where the UE can simultaneously communicate with all the S_j's. For convenience, we may assume that there is a virtual and imaginary server S'_0, that is responsible for all wireless communications from the S_0 to all the S_j's, where $1 \le j \le m$.

We would like to mention that for an ES_j, the wireless communication (done by S'_0) and the computation (done by ES_j) may not be consecutive, i.e., there could be some time delay due to the unavailability of ES_j. Similarly, for a CS_j, the wireless communication (done by S'_0), the wired communication (done by CF_j), and the computation (done by CS_j) may not be consecutive.

For an ES_j, the wireless communication of one task T_i can overlap with the computation of another task $T_{i'}$, i.e., while T_i is transmitted to ES_j, ES_j is computing $T_{i'}$. For a CS_j, the wireless communication of one task T_i can overlap with the wired communication and computation of another task $T_{i'}$, i.e., while T_i is transmitted to CF_j, $T_{i'}$ is transmitted to CS_j and to be computed by CS_j. This is different from and more difficult than [16,17], where a server S_j must receive all tasks assigned to S_j, and then start to execute these tasks.

2.4 Model Extensibility

We would like to mention that our models in this paper can be extended easily to accommodate more sophisticated DAG execution situations.

The first issue is inter-task communication between a predecessor task T_i and and a successor task $T_{i'}$, where $T_i \prec T_{i'}$. This can happen when $T_{i'}$ needs the results generated by T_i. We consider several different cases. (1) If both T_i and $T_{i'}$ are executed on the same S_j, there is no data communication time. (2) If T_i and $T_{i'}$ are executed on S_j and $S_{j'}$ respectively with $j, j' \ge 1$, data communication can be handled by wired communication and the communication time can be converted to computation time, i.e., we can treat $r_{j'}$ to be certain increased amount to equivalently include the wired communication time from S_j and $S_{j'}$. (3) If T_i is executed on S_j with $j \ge 1$ and $T_{i'}$ is executed on S_0, data communication can be handled by wireless communication; however, S_0 receives data instead of transmitting data. Hence, the communication time can again be converted to computation time. (4) If T_i is executed on S_0 and $T_{i'}$ is executed on S_j with $j \ge 1$, data communication should be handled by wireless communication and S_0 transmits data. In this case, the communication time can be added to the wireless communication time of $T_{i'}$, i.e., we can treat $d_{j'}$ to be certain increased amount to include the wireless communication time from S_0 and S_j.

The second issue is output data collection. Suppose $\mathscr{T}'$ is the task set that generates final results (i.e., output data) which should be collected together. To handle this issue, we can add a dummy task T_{n+1} with $d_{n+1} = r_{n+1} = 0$ and $T_i \prec T_{n+1}$ for all $T_i \in \mathscr{T}'$. Of course, both d_{n+1} and r_{n+1} might be adjusted based on the above discussion of inter-task communication.

As can be seen later, our algorithms make scheduling decisions solely based on task readiness and server availability and do not rely on the r_i's and the d_i's, the above adjustments of r_i and d_i do not affect our algorithms at all.

3 DAG Scheduling

In this section, we consider DAG scheduling on edge and cloud servers. We define our DAG-scheduling problem, present a motivational example, develop our heuristic algorithms, analyze the time complexity, derive a lower bound for the optimal schedule length, and experimentally evaluate the performance of our heuristic algorithms.

3.1 Problem Definition

In this section, we define our DAG-scheduling problem.

A *schedule* determines when and where to execute T_i, including wireless communication, wired communication, and computation, for all $1 \leq i \leq n$. The *makespan* is the time when all servers finish their computations.

Our DAG scheduling problem in this paper can be described as follows.

Problem 1: DAG Scheduling on Edge and Cloud Servers.
Input: Servers $S_0, S_1, S_2, ..., S_m$, and a DAG $G = (\mathscr{T}, \prec)$.
Output: A schedule of G on $S_0, S_1, S_2, ..., S_m$ with the minimum makespan.

The above problem is NP-hard even for the following extreme case: (1) tasks are independent, i.e., $\prec = \emptyset$; (2) there is no communication cost, i.e., $d_i = 0$ for all $1 \leq i \leq n$; (3) there is only one ES, i.e., $m = 1$; (4) $s_0 = s_1$. In this simple case, the classic *partition problem* ([6], p. 47) can be reduced to our problem, in the sense that the set $\{r_1, r_2, ..., r_n\}$ has a partition if and only if the makespan is $0.5(r_1 + r_2 + \cdots + r_n)$.

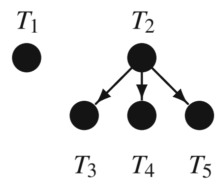
Due to the NP-hardness of the problem, we are only able to produce heuristic solutions. Therefore, there is a challenging issue of comparing a heuristic solution with an optimal solution.

3.2 The DSECS-H Algorithm

In this section, we present a motivational example, develop our heuristic algorithms, and analyze the time complexity.

3.2.1 A Motivational Example

Let us consider S_0, S_1, S_2, where S_1 is an ES and S_2 is a CS. Assume that we have $G = (\mathscr{T}, \prec)$, where $\mathscr{T} = \{T_1, T_2, T_3, T_4, T_5\}$, and $T_2 \prec T_3$, $T_2 \prec T_4$, $T_2 \prec T_5$, as shown below:



A sample schedule is demonstrated in Fig. 2. Each box stands for a communication or computation activity. T_1 is executed on S_0. T_2, T_3, T_5 are offloaded to S_1. T_4 is offloaded to S_2. S_0' is responsible for all wireless communications.

Notice that the wireless transmission of T_3 to S_1 overlaps with the computation of T_2 on S_1, and the wireless transmission of T_5 to S_1 overlaps with the computation of T_3 on S_1. After T_3 is transmitted to S_1, the computation of T_3 is delayed because S_1 is processing T_2. S_1 becomes idle after T_3 is completed, since the transmission of T_5 is not completed yet. The processing of T_4 on S_2 requires wired communication and computation.

3.2.2 Heuristic Algorithms

Our algorithm is called DSECS-H (DAG Scheduling on Edge and Cloud Servers with Heuristic H). Since H can vary, DSECS-H is essentially a class of task-scheduling algorithms.

Our greedy algorithms (see Algorithm 1) are based on the classic list-scheduling algorithm [7].

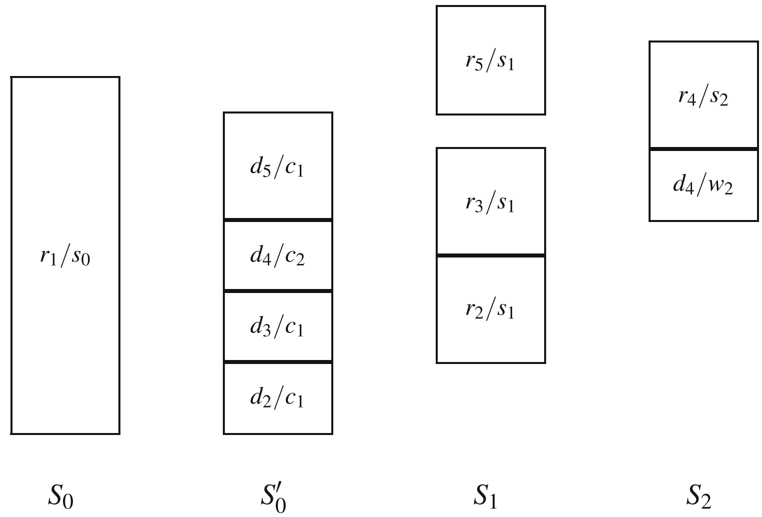


Fig. 2 Illustration of the motivational example

Initially, tasks in $\mathscr{T}$ are arranged to a list L according to some heuristic H (line (1)).

Let CompRT_j be the computation ready time of S_j, i.e., the time when S_j is ready for the next computation if S_j is the UE or an ES, and for the next wired communication plus computation if S_j is a CS, where $0 \le j \le m$. Initially, $\text{CompRT}_j = 0$, for all $0 \le j \le m$ (lines (2)–(4)).

Let CommRT be the wireless communication ready time, i.e., the time when S'_0 is ready for the next wireless communication. Initially, CommRT = 0 (line (5)).

A global clock is used, which is set to 0 initially (line (6)).

Each repetition of the while-loop (lines (7)–(36)) schedules one ready task on one available server.

A task is ready to be scheduled if all its predecessors have been finished. Lines (8)–(17) find a ready task. The first ready task T_i in L is chosen (line (16)) and removed from L (line (17)). If there is no ready task due to precedence constraints, we will have to wait for some currently running tasks to be finished (lines (8)–(15)). Each repetition of the while-loop (lines (8)–(15)) only waits for one finished task. The loop is repeated until there is a ready task (line (8)). In each repetition, we find

$$j = \text{argmin}_{S_j \text{ is computing}}\{\text{CompRT}_j\}$$

(line (9)), i.e., S_j is the first server to finish its task. The clock is set to be CompRT_j (line (10)). Assume that S_j is processing T_i (line (11)). Then, T_i is the first currently running task to be finished. Hence, all precedence constraints between T_i and its successors $T_{i'}$, i.e., $T_i \prec T_{i'}$, are released (lines (12)–(14)).

A server S_j is available if $\text{CompRT}_j \le$ clock. Lines (18)–(26) find an available server. If there are several available servers, the first one is chosen (line (18)). If there is no available server, we will have to wait for some currently running tasks to be finished (lines (19)–(26)). Lines (20)–(25) are actually identical to lines (9)–(14).

The ready task T_i is scheduled on the available server S_j at clock (lines (27)–(35)). If S_j is the UE (line (27)), we set CompRT_0 = clock + r_i/s_0 (line (28)). If S_j is an ES (line (29)), we set CommRT = CommRT + d_i/c_j (line (30)) and $\text{CompRT}_j = \max\{\text{CommRT}, \text{clock}\} + r_i/s_j$ (line (31)). If S_j is a CS (line (32)), we set CommRT = CommRT + d_i/c_j (line (33)) and $\text{CompRT}_j = \max\{\text{CommRT}, \text{clock}\} + d_i/w_j + r_i/s_j$ (line (34)). Notice that lines (30) and (33) guarantee sequential wireless communication, and lines (31) and (34) allow overlapped communication and computation. Also, lines (31) and (34) mean that if wireless communication takes such a long time that CommRT > clock, T_i is scheduled at time CommRT, not clock. This causes S_j to wait for some amount of time.

When the while-loop in lines (7)–(36) is completed, we have the following makespan of G:

$$\text{makespan}(G) = \max_{0 \le j \le m}\{\text{CompRT}_j\}.$$

Algorithm 1: DAG Scheduling on Edge and Cloud Servers with Heuristic H (DSECS-H)

Input: Servers $S_0, S_1, S_2, ..., S_m$, and a DAG $G = (\mathscr{T}, \prec)$.
Output: A schedule of G on $S_0, S_1, S_2, ..., S_m$ with the minimum makespan.

---

make a list L of tasks by using heuristic H; //heuristics (1)
for ($j \leftarrow 0$; $j \le m$; j++) **do** (2)
  $\text{CompRT}_j \leftarrow 0$; (3)
end do; (4)
$\text{CommRT} \leftarrow 0$; //to guarantee sequential wireless communication (5)
clock $\leftarrow 0$; //global clock (6)
while (L is not empty) **do** //each repetition schedules one task (7)
  //choose a ready task T_i
  while (there is no ready task) **do** //precedence constraint (8)
    $j \leftarrow \text{argmin}_{S_j \text{ is computing}}\{\text{CompRT}_j\}$; //$S_j$ is the next server to complete a task (9)
    clock $\leftarrow \text{CompRT}_j$; (10)
    $T_i \leftarrow$ the task just finished on S_j; (11)
    for (each successor $T_{i'}$ of T_i) **do** (12)
      release the precedence constraint $T_i \prec T_{i'}$; (13)
    end do; (14)
  end do; (15)
  $T_i \leftarrow$ the first ready task in L; //the first ready task (16)
  remove T_i from L; (17)
  //choose an available server S_j
  $j \leftarrow$ the smallest j' such that $\text{CompRT}_{j'} \le$ clock; //the first available server (18)
  if (j is not found) (19)
    $j \leftarrow \text{argmin}_{S_j \text{ is computing}}\{\text{CompRT}_j\}$; //$S_j$ is the next server to complete a task (20)
    clock $\leftarrow \text{CompRT}_j$; //$S_j$ is now available (21)
    $T_i \leftarrow$ the task just finished on S_j; (22)
    for (each successor $T_{i'}$ of T_i) **do** (23)
      release the precedence constraint $T_i \prec T_{i'}$; (24)
    end do; (25)
  end if; (26)
  //schedule T_i on S_j at clock
  if ($j = 0$) //S_j is the UE (27)
    $\text{CompRT}_0 \leftarrow \text{clock} + r_i/s_0$; (28)
  else if (S_j is an ES) (29)
    $\text{CommRT} \leftarrow \text{CommRT} + d_i/c_j$; //sequential wireless communication (30)
    $\text{CompRT}_j \leftarrow \max\{\text{CommRT}, \text{clock}\} + r_i/s_j$; //overlapped comm and comp (31)
  else //S_j is a CS (32)
    $\text{CommRT} \leftarrow \text{CommRT} + d_i/c_j$; //sequential wireless communication (33)
    $\text{CompRT}_j \leftarrow \max\{\text{CommRT}, \text{clock}\} + d_i/w_j + r_i/s_j$; //overlapped comm and comp (34)
  end if; (35)
end do. (36)

3.2.3 Time Complexity

The time complexity of the DSECS-H algorithm is analyzed as follows.

Line (1) takes $O(n \log n)$ time.

Lines (2)–(4) take $O(m)$ time.

Line (9) takes $O(m)$ time. Since line (9) is executed at most n times, the overall time to execute line (9) is $O(mn)$. The overall time to execute lines (12)–(14) is $O(n^2)$. Hence, the overall time to execute lines (8)–(15) is $O(mn + n^2)$.

Line (18) takes $O(m)$ time. Since line (18) is executed at most n times, the overall time to execute line (18) is $O(mn)$. Line (20) takes $O(m)$ time. Since line (20) is executed at most n times, the overall time to execute line (20) is $O(mn)$. The overall time to execute lines (23)–(25) is $O(n^2)$. Hence, the overall time to execute lines (18)–(26) is $O(mn + n^2)$.

Lines (27)–(35) take $O(1)$ time. Since lines (27)–(35) is executed n times, the overall time to execute lines (27)–(35) is $O(n)$.

To summarize, the time complexity of the DSECS-H algorithm is $O(mn + n^2)$.

3.3 A Lower Bound

In this section, we derive a lower bound for the optimal schedule length, i.e., $\text{makespan}^*(G)$.

Let $\gamma = \min_{1 \le i \le n}\{d_i/r_i\}$. We consider $G' = (\mathscr{T}', \emptyset)$, where $\mathscr{T}' = \{T_1', T_2', ..., T_n'\}$, with $d_i = \gamma r_i$, for all $1 \le i \le n$. It is clear that each task T_i' in G' has less amount of communication than T_i in G; furthermore, G' does not have any precedence constraint. Therefore, we have

$$\text{makespan}^*(G) \ge \text{makespan}^*(G').$$

For each server S_j, define R_j to be the total amount of computation on S_j, $D_j = \gamma R_j$ to be the total amount of communication of tasks processed on S_j, and W_j to be the total waiting time of S_j, where $0 \le j \le m$.

For a task T_i computed on CS_j, we have

$$\begin{aligned} d_i/w_j + r_i/s_j &= \gamma r_i/w_j + r_i/s_j = r_i(\gamma/w_j + 1/s_j) \\ &= r_i/(w_j s_j/(\gamma s_j + w_j)) \\ &= r_i/(s_j/(1 + \gamma s_j/w_j)). \end{aligned}$$

Hence, we can treat

$$s_j \leftarrow s_j/(1 + \gamma s_j/w_j)$$

as the *effective computation speed* of CS_j. This way, all ES and CS can be unified.

It is clear that

$$\text{makespan}^*(G') = \max\left\{\text{CompRT}_0, \text{CompRT}_1, ..., \text{CompRT}_m, \text{CommRT}\right\},$$

where

$$\text{CompRT}_j = R_j/s_j + W_j \ge R_j/s_j,$$

for all $0 \le j \le m$. Also, we have

$$\begin{aligned}\text{CommRT} &= D_1/c_1 + D_2/c_2 + \cdots + D_m/c_m \\ &= \gamma(R_1/c_1 + R_2/c_2 + \cdots + R_m/c_m).\end{aligned}$$

Consequently, we get

$$\text{makespan}^*(G') \ge \max\{R_0/s_0, R_1/s_1, R_2/s_2, ..., R_m/s_m, \gamma(R_1/c_1 + R_2/c_2 + \cdots + R_m/c_m)\}.$$

Let B be defined as follows:

$$B = \max\{R_0/s_0, R_1/s_1, R_2/s_2, ..., R_m/s_m, \gamma(R_1/c_1 + R_2/c_2 + \cdots + R_m/c_m)\},$$

which is treated as a function of $R_0, R_1, R_2, ..., R_m$. The above discussion implies that the minimum value of B over different choices of $R_0, R_1, R_2, ..., R_m$ can be a lower bound for the optimal schedule length $\text{makespan}^*(G')$ (and $\text{makespan}^*(G)$ as well).

Let $R = R_0 + R_1 + R_2 + \cdots + R_m$ be the total amount of computation, and $S = s_0 + s_1 + s_2 + \cdots + s_m$ be the aggregated computation speed of a device-edge-cloud collaborative computing system.

We need to minimize B, subject to the condition $R_0 + R_1 + R_2 + \cdots + R_m = R$. Note that since we are finding a lower bound, we will treat this as a pure numerical optimization problem, whose solution may not be realized by any real schedule.

To this end, we let

$$R_0/s_0 = R_1/s_1 = R_2/s_2 = \cdots = R_m/s_m = \tau,$$

for some τ. (Notice that this does not guarantee the minimization of B yet if the wireless communication cost is too high, as shown below.) Then, we have $R_j = s_j\tau$, which gives $R = (s_0 + s_1 + s_2 + \cdots + s_m)\tau = S\tau$, $\tau = R/S$, and $R_j = (s_j/S)R$, for all $0 \le j \le m$.

Therefore, we have

$$B \ge \max\{R/S, \gamma((s_1/S)(R/c_1) + (s_2/S)(R/c_2) + \cdots + (s_m/S)(R/c_m))\},$$

that is,

$$B \ge (R/S)\max\{1, \gamma(s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m)\}.$$

Notice that $\gamma(s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m)$ stands for the amount of wireless communication on S_0'.

If $\gamma(s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m) \le 1$ (Fig. 3(a)), then $B \ge R/S$.

If $\gamma(s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m) > 1$ (Fig. 3(b)), we need to move some tasks to the UE (Fig. 3(c)). For the same amount of workload moved to the UE, we try to maximize the reduction of $R_1/c_1 + R_2/c_2 + \cdots + R_m/c_m$. Therefore, we move tasks from those S_j's with small c_j's. Without loss of generality, we assume that $c_1 \le c_2 \le \cdots \le c_m$. Let $R' = R_1 + \cdots + R_{k-1} + R_k'$ be the amount of workload moved to the UE, such that

$$(R_0 + R')/s_0 = \gamma((R_k - R_k')/c_k + R_{k+1}/c_{k+1} + \cdots + R_m/c_m).$$

In the above identity, the left-hand side is the increased computation time of the UE, and the right-hand side is the reduced wireless communication time of S_0'. The index k is determined such that

$$(R_0 + R_1 + \cdots + R_{k-1})/s_0 < \gamma(R_k/c_k + R_{k+1}/c_{k+1} + \cdots + R_m/c_m),$$

and

$$(R_0 + R_1 + \cdots + R_{k-1} + R_k)/s_0 \ge \gamma(R_{k+1}/c_{k+1} + \cdots + R_m/c_m).$$

Since

$$(R_0 + R_1 + \cdots + R_{k-1} + R_k')/s_0 = \gamma((R_k - R_k')/c_k + \cdots + R_m/c_m),$$

we get

$$(1 + \gamma s_0/c_k)R_k' = \gamma s_0(R_k/c_k + \cdots + R_m/c_m) - (R_0 + R_1 + \cdots + R_{k-1}),$$

which implies that

$$R_k' = (\gamma s_0(R_k/c_k + \cdots + R_m/c_m) - (R_0 + R_1 + \cdots + R_{k-1}))/(1 + \gamma s_0/c_k).$$

Then, we obtain

$$B \ge (R_0 + R')/s_0.$$

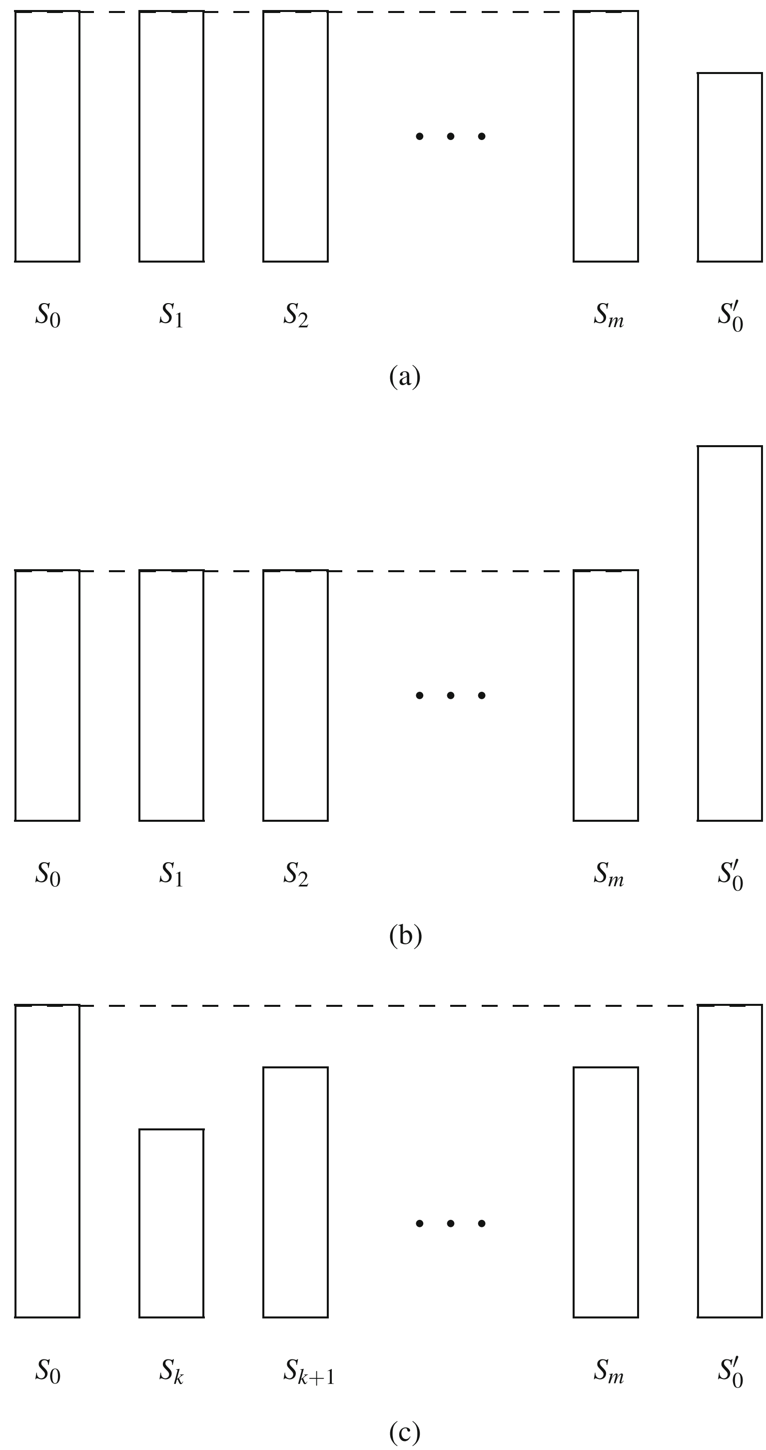


Fig. 3 Illustration of the lower bound

Notice that R' is defined in such a way that for the same amount of increase to CompRT_0, CommRT is decreased for the most amount. Also, R_0 cannot be reduced, since any reduction means that more tasks are assigned to some S_j, $1 \le j \le n$, and CommRT will be increased.

The above discussion can be summarized as follows.

Theorem 1 *We have the following lower bound for the optimal makespan. If* $\gamma(s_1/c_1+s_2/c_2+\cdots+s_m/c_m) \le 1$*, then*

$$\mathrm{makespan}^*(G) \ge R/S.$$

If $\gamma(s_1/c_1+s_2/c_2+\cdots+s_m/c_m) > 1$*, then*

$$\mathrm{makespan}^*(G) \ge (R_0 + R')/s_0.$$

We use $\bar{B}$ to denote the lower bound in Theorem 1.

3.4 Performance Evaluation

In this section, we experimentally evaluate the performance of our heuristic algorithms.

3.4.1 Parameter Settings

We consider a device-edge-cloud collaborative computing system with one UE (i.e., S_0), $m_1 = 4$ ES (i.e., S_1, S_2, S_3, S_4), and $m_2 = 2$ CS (i.e., S_5, S_6). The computation speeds s_j, wireless communication speeds c_j, and wired communication speeds w_j are given below. These (and other) parameters are chosen based on the current computation and communication technologies.

	S_0	S_1	S_2	S_3	S_4	S_5	S_6
s_j (Bips)	1.5	2.8	2.7	2.6	2.5	3.5	3.6
c_j (Mbps)		35	40	45	50	55	60
w_j (Mbps)						95	90

The computation requirements (i.e., the r_i's) are independent and identically distributed (i.i.d.) random variables uniformly distributed in the range [1.5, 5.0] GI. The communication requirement is $d_i = \gamma_i r_i$ in the range [1.5, 25.0] MB, where the γ_i's are i.i.d. random variables uniformly distributed in the range $[\gamma, 5.0]$ MB/GI. We set $\gamma = 1.0$ for computation-intensive tasks and $\gamma = 3.0$ for communication-intensive tasks.

A DAG is a random graph, where the existence of arcs $(T_i, T_{i'})$ with $i < i'$ are independent of each other with identical probability p.

3.4.2 Experimental Results

There are several heuristics H, e.g., H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50 [13,15]. These heuristics are: *Original Order* (ORG), *Smallest Requirement First* (SRF), *Largest Requirement First* (LRF), *Smallest Data First* (SDF), *Largest Data First* (LDF), *Smallest Requirement-Data-Ratio First* (SRD), *Largest Requirement-Data-Ratio First* (LRD), *Best of k Random Orders* (RANk).

It is clear that

$$\mathrm{makespan}(G)/\mathrm{makespan}^*(G) \le \mathrm{makespan}(G)/\bar{B}.$$

Thus, the expectation of the ratio $\mathrm{makespan}(G)/\bar{B}$, i.e., $\boldsymbol{E}(\mathrm{makespan}(G)/\bar{B})$, can be considered as an *expected performance bound* of our heuristic algorithms. For given n and H, the expected performance bound can be obtained experimentally as follows. We generate M random DAGs: $G_1, G_2, ..., G_M$. For each G_k, we run algorithm DSECS-H, get its $\mathrm{makespan}(G_k)$, calculate the lower bound $\bar{B}$, and record the ratio $\mathrm{makespan}(G_k)/\bar{B}$. The average of the M ratios is returned as the expected performance bound. We set $M = 2,000$ for all experiments.

In Table 1, for each combination of n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for sparse DAG ($p = 2/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is ±1.52179%.

In Table 2, for each combination of n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for sparse DAG ($p = 2/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is ±1.27554%.

In Table 3, for each combination of n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for dense DAG ($p = 5/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is ±1.43607%.

Table 1 Expected Performance Bound for Sparse DAG with Computation-Intensive Tasks ($p = 2/n$, $\gamma = 1.0$, 99% C.I. = ±1.52179%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	2.40280	2.40300	2.41920	2.38099	2.42418	2.43530	2.39627	1.97798	1.90146	1.87509
40	1.55366	1.69899	1.68168	1.68101	1.70558	1.71947	1.66473	1.42007	1.38201	1.36777
60	1.35547	1.49587	1.47842	1.48081	1.50139	1.51907	1.46901	1.29204	1.26716	1.25731
80	1.27979	1.39832	1.37902	1.38368	1.40350	1.42210	1.37393	1.23858	1.22225	1.21503
100	1.24157	1.33830	1.32517	1.32714	1.35164	1.37834	1.32170	1.21144	1.19727	1.19146
120	1.21775	1.30361	1.28903	1.29302	1.31621	1.33526	1.28640	1.19281	1.18166	1.17666
140	1.20252	1.27577	1.26687	1.26833	1.29099	1.30929	1.26322	1.18149	1.17203	1.16769
160	1.18892	1.25501	1.24347	1.24782	1.27442	1.29152	1.24345	1.17113	1.16270	1.15895
180	1.17978	1.23998	1.23344	1.23339	1.25912	1.27781	1.22875	1.16500	1.15691	1.15344
200	1.17356	1.22838	1.21863	1.22273	1.24477	1.26663	1.21720	1.15967	1.15244	1.14918

Table 2 Expected Performance Bound for Sparse DAG with Communication-Intensive Tasks ($p = 2/n$, $\gamma = 3.0$, 99% C.I. = ±1.27554%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	2.32740	2.31583	2.34526	2.30668	2.33654	2.32589	2.34218	1.97437	1.91137	1.89149
40	1.64170	1.74415	1.74220	1.74578	1.74143	1.74421	1.74187	1.53458	1.50490	1.49316
60	1.48540	1.58975	1.58348	1.58815	1.57966	1.58453	1.58547	1.43391	1.41660	1.41007
80	1.42583	1.51809	1.51212	1.51237	1.51098	1.51292	1.50522	1.39414	1.38241	1.37741
100	1.39858	1.47549	1.46472	1.47240	1.46381	1.46971	1.46908	1.37425	1.36416	1.36029
120	1.37699	1.44220	1.43805	1.43937	1.43535	1.43805	1.43787	1.35957	1.35190	1.34878
140	1.36606	1.42318	1.41899	1.42589	1.41631	1.41729	1.41921	1.35125	1.34390	1.34124
160	1.35619	1.40881	1.40448	1.40645	1.40147	1.40337	1.40096	1.34367	1.33777	1.33503
180	1.35047	1.39414	1.39227	1.39442	1.39103	1.39244	1.39194	1.33935	1.33385	1.33131
200	1.34388	1.38526	1.38359	1.38502	1.38207	1.38214	1.38111	1.33453	1.32945	1.32732

Table 3 Expected Performance Bound for Dense DAG with Computation-Intensive Tasks ($p = 5/n$, $\gamma = 1.0$, 99% C.I. = ±1.43607%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	4.13048	3.74695	3.94916	3.80075	3.91101	3.82846	3.87072	3.25277	3.14199	3.11207
40	2.29334	2.14749	2.23183	2.16350	2.21291	2.18398	2.18565	1.87111	1.80926	1.78897
60	1.68262	1.67141	1.70060	1.68345	1.69448	1.69175	1.68101	1.47872	1.44075	1.42675
80	1.42250	1.48363	1.48912	1.48638	1.49165	1.49007	1.48369	1.32650	1.30213	1.29285
100	1.30204	1.38599	1.38526	1.37867	1.38774	1.39291	1.38195	1.25342	1.23591	1.22911
120	1.25594	1.33351	1.33284	1.33490	1.33575	1.34037	1.32856	1.22118	1.20809	1.20262
140	1.22516	1.29724	1.29583	1.29405	1.29354	1.30212	1.29038	1.19746	1.18653	1.18194
160	1.20469	1.27144	1.26657	1.26792	1.27320	1.27586	1.26538	1.18265	1.17301	1.16899
180	1.19155	1.25340	1.25070	1.24894	1.25254	1.25683	1.24738	1.17315	1.16468	1.16114
200	1.18192	1.23408	1.23076	1.23242	1.23479	1.24159	1.23065	1.16469	1.15718	1.15381

In Table 4, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for dense DAG ($p = 5/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 1.42009\%$.

Since the value of $s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m$ is 0.37435, we have $\gamma(s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m) < 1$ when $\gamma = 1.0$, and $\gamma(s_1/c_1 + s_2/c_2 + \cdots + s_m/c_m) > 1$ when $\gamma = 3.0$.

We have the following observations.

First, all our heuristic algorithms can produce near-optimal schedules whose makespans are very close to the optimal makespans, in the sense that the expected performance bounds are very close to one, especially when n gets large. This means that our heuristic algorithms are able to efficiently utilize all communication and computation resources in processing both computation-intensive and communication-intensive tasks with both sparse and dense precedence constraints on a device-edge-cloud collaborative computing system.

Second, the simple ORG strategy seems to perform better than SRF, LRF, SDF, LDF, SRD, and LRD, except for small n and dense DAG. This means that all these sorting methods are not really productive. This phenomenon has been observed in other studies (e.g., [15]), primarily due to precedence constraints. Furthermore, RAN10, RAN30, and RAN50 can further improve the scheduling performance by spending more execution time.

4 Energy Aspects of Algorithm DSECS-*H*

In this section, we consider the energy aspects of our heuristic algorithms. We describe the power consumption models and discuss the energy consumption and energy efficiency of algorithm DSECS-H.

4.1 Power Consumption Models

In this section, we describe the power consumption models for both computation and communication.

The computation power consumption model of the UE is $P_0(s_0) = \xi s_0^\alpha + P_s$ (measured by Watts), where ξ (measured by Watts/Bips$^\alpha$) and α are technology-dependent constants, and P_s (measured by Watts) is the static component of power consumption.

The power consumption model of the wireless communication channel between the UE and S_j (ES$_j$ or CS$_j$) is $P_j(c_j) = (2^{c_j/b_j} - 1)/\beta_j$ (measured by Watts), for all $1 \le j \le m$, where b_j is the channel bandwidth (measured by Mbps) and β_j (measured by Watts^{-1}) is a combined quantity of several factors such as background noise power, interference on the communication channel, and channel gain [13–17].

If task T_i is executed locally on the UE with computation speed $s_{0,i}$, the computation energy consumption of T_i is $E_i = P_0(s_{0,i})(r_i/s_{0,i}) = ((\xi s_{0,i}^\alpha + P_s)/s_{0,i})r_i$ (measured by Joules).

If task T_i is executed remotely on S_j (ES$_j$ or CS$_j$) with wireless communication speed $c_{j,i}$, where $1 \le j \le m$, the communication energy consumption of T_i

Table 4 Expected Performance Bound for Dense DAG with Communication-Intensive Tasks ($p = 5/n$, $\gamma = 3.0$, 99% C.I. = $\pm 1.42009\%$)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	3.74595	3.43154	3.62118	3.44106	3.60525	3.51841	3.52553	2.98030	2.87563	2.84741
40	2.17508	2.06998	2.14115	2.07026	2.13674	2.10185	2.10265	1.83548	1.78787	1.77207
60	1.69552	1.69051	1.71521	1.69586	1.71280	1.69982	1.69947	1.55267	1.52449	1.51493
80	1.52744	1.56268	1.57657	1.56415	1.56998	1.56557	1.56713	1.45240	1.43493	1.42838
100	1.44494	1.49864	1.50134	1.49507	1.49745	1.49716	1.49969	1.40383	1.39166	1.38732
120	1.40567	1.45834	1.46167	1.46113	1.46066	1.45796	1.45776	1.37839	1.36792	1.36424
140	1.38555	1.43451	1.43407	1.43425	1.43184	1.43224	1.43422	1.36267	1.35470	1.35140
160	1.36875	1.41733	1.41575	1.41562	1.41290	1.41249	1.41433	1.35168	1.34511	1.34230
180	1.35869	1.40212	1.40051	1.40043	1.40120	1.40029	1.40200	1.34413	1.33841	1.33614
200	1.35216	1.39179	1.38939	1.39198	1.39174	1.39002	1.39076	1.33930	1.33407	1.33175

is $E_i = P_j(c_{j,i})(d_i/c_{j,i}) = ((2^{c_{j,i}/b_j} - 1)/(\beta_j c_{j,i}))d_i$ (measured by Joules).

Note that to execute T_i, either $s_{0,i}$ or $c_{j,i}$ needs to be decided.

The total energy consumption of G is

$$\text{energy}(G) = \sum_{1 \le i \le n} E_i,$$

which is actually the cost measure of task execution in a device-edge-cloud collaborative computing system.

4.2 Energy Consumption of Algorithm DSECS-H

In this section, we discuss the energy consumption of algorithm DSECS-H.

We keep the same parameter settings of Section 3.4, and have the following additional parameter settings for our power consumption models. For the computation power consumption model of the UE, we have $\xi = 0.1$ Watts/Bips^2, $\alpha = 2.0$, and $P_s = 0.05$ Watts. For the wireless communication power consumption model, we have the following.

	S_1	S_2	S_3	S_4	S_5	S_6
b_j (Mbps)	30	31	32	33	34	35
β_j (1/Watts)	2.00	1.95	1.90	1.85	1.80	1.75

For given n and H, the expected energy consumption $\boldsymbol{E}(\text{energy}(G))$ can be obtained experimentally as follows. We generate M random DAGs: $G_1, G_2, ..., G_M$. For each G_k, we run algorithm DSECS-H and record its energy(G_k). The average of the M values is returned as the expected energy consumption. We set $M = 2,000$ for all experiments.

The following experimental data demonstrate the expected energy consumption of algorithm DSECS-H.

In Table 5, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected energy consumption for sparse DAG ($p = 2/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is $\pm 0.70704\%$.

In Table 6, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected energy consumption for sparse DAG ($p = 2/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 0.51276\%$.

In Table 7, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected energy consumption for dense DAG ($p = 5/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is $\pm 0.78192\%$.

In Table 8, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected energy consumption for dense DAG ($p = 5/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 0.62684\%$.

We have the following observations. First, all our heuristic algorithms consume about the same amount of energy. Second, for a DAG with n computation-intensive tasks, energy(G) is approximately $0.235n$; and for a DAG with n communication-intensive tasks energy(G) is approximately $0.3n$.

4.3 Energy Efficiency of Algorithm DSECS-H

In this section, we discuss the energy efficiency of algorithm DSECS-H.

Although the algorithm DSECS-H is not designed for energy-constrained scheduling. we can still discuss its energy efficiency as follows.

For a DAG G, let makespan(G) and energy(G) be the schedule length and energy consumption of algorithm DSECS-H. For the same amount of energy(G), let makespan$^*(G)$ be the optimal makespan. Then, the ratio

$$\text{efficiency}(G) = \text{makespan}^*(G)/\text{makespan}(G)$$

can be considered as the *energy efficiency* of algorithm DSECS-H.

It is clear that

$$\text{makespan}^*(G)/\text{makespan}(G) \ge \tilde{B}/\text{makespan}(G),$$

where $\tilde{B}$ (to be derived in Section 5.3) is a lower bound for the optimal schedule length with the same amount energy(G) of energy consumption. Thus, the expectation of the ratio $\tilde{B}/\text{makespan}(G)$, i.e.,

Table 5 Energy Consumption of Algorithm DSECS-H for Sparse DAG with Computation-Intensive Tasks ($p = 2/n$, $\gamma = 1.0$, 99% C.I. = ±0.70704%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	5.25690	5.20378	5.25063	5.28155	5.09136	5.08201	5.35309	4.81817	4.72339	4.68742
40	9.57491	9.69376	9.67312	9.73800	9.58115	9.62126	9.79132	9.23099	9.11750	9.07257
60	14.04051	14.23726	14.23624	14.26279	14.14735	14.17814	14.30262	13.71243	13.58984	13.53962
80	18.63047	18.86624	18.82656	18.88147	18.75787	18.82186	18.91034	18.29237	18.15751	18.10301
100	23.16895	23.44626	23.38256	23.45007	23.33332	23.38686	23.45066	22.80907	22.66670	22.60476
120	27.68007	27.93240	27.89900	27.94640	27.85031	27.91281	27.94712	27.30382	27.14151	27.07647
140	32.27595	32.54579	32.50402	32.52563	32.45330	32.51684	32.53156	31.87076	31.69856	31.62714
160	36.86984	37.16050	37.08973	37.12506	37.07385	37.15664	37.13822	36.45239	36.26689	36.20269
180	41.48573	41.76185	41.72579	41.71689	41.69105	41.78362	41.74044	41.03490	40.85207	40.77478
200	46.02644	46.30710	46.26708	46.27677	46.25526	46.34147	46.25825	45.55519	45.36210	45.28757

Table 6 Energy Consumption of Algorithm DSECS-H for Sparse DAG with Communication-Intensive Tasks ($p = 2/n$, $\gamma = 3.0$, 99% C.I. = ±0.51276%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	6.32859	6.29596	6.32187	6.31170	6.26748	6.25472	6.38032	6.00163	5.92675	5.90140
40	12.12313	12.20325	12.19612	12.22237	12.14284	12.16865	12.25481	11.86329	11.78331	11.75289
60	17.93047	18.04962	18.03923	18.05953	17.98085	18.00466	18.08178	17.68269	17.60116	17.56777
80	23.83000	23.99710	23.96880	24.00202	23.93057	23.93659	24.02149	23.60140	23.51599	23.47969
100	29.74313	29.90316	29.88353	29.91801	29.84154	29.84910	29.92699	29.51154	29.41797	29.37956
120	35.67137	35.83432	35.81745	35.85008	35.77973	35.78401	35.85664	35.42654	35.32630	35.28631
140	41.55752	41.72418	41.71850	41.75721	41.65944	41.66820	41.75939	41.31225	41.20842	41.16508
160	47.46421	47.63089	47.61243	47.63938	47.57163	47.56408	47.64953	47.19192	47.08193	47.04013
180	53.37779	53.54649	53.52223	53.55921	53.48947	53.48394	53.58070	53.10807	52.99688	52.94526
200	59.33916	59.51665	59.48565	59.52784	59.45357	59.43591	59.51429	59.04968	58.93568	58.88864

Table 7 Energy Consumption of Algorithm DSECS-H for Dense DAG with Computation-Intensive Tasks ($p = 5/n$, $\gamma = 1.0$, 99% C.I. = ±0.78192%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	6.22141	5.87902	6.09661	6.08055	5.88169	5.78615	6.17188	5.46866	5.35231	5.31870
40	10.39848	10.13468	10.27899	10.33913	10.03454	9.99365	10.46098	9.66102	9.52389	9.47796
60	14.57401	14.50417	14.57730	14.64539	14.37570	14.35937	14.72364	13.98255	13.83381	13.77647
80	18.92579	18.98543	19.04309	19.09346	18.83966	18.82468	19.18381	18.43436	18.28134	18.22172
100	23.33028	23.50968	23.52339	23.60930	23.35706	23.36066	23.66671	22.90105	22.74553	22.68302
120	27.82300	28.03515	28.06074	28.13328	27.90152	27.90260	28.19157	27.40692	27.25102	27.17940
140	32.40266	32.59326	32.60978	32.68193	32.48492	32.47779	32.73817	31.95051	31.77977	31.71364
160	36.99663	37.21547	37.23352	37.29318	37.08866	37.12240	37.33818	36.53723	36.35504	36.28698
180	41.54582	41.79528	41.78088	41.83182	41.63435	41.67551	41.90106	41.08176	40.89632	40.80867
200	46.06396	46.33474	46.33047	46.34653	46.17278	46.19072	46.39667	45.57148	45.36910	45.29252

Table 8 Energy Consumption of Algorithm DSECS-H for Dense DAG with Communication-Intensive Tasks ($p = 5/n$, $\gamma = 3.0$, 99% C.I. = ±0.62684%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	7.03908	6.75713	6.92223	6.79541	6.86775	6.74655	6.92914	6.42200	6.33290	6.30835
40	12.47435	12.30171	12.39915	12.32913	12.31973	12.23055	12.47254	11.94227	11.83508	11.79833
60	18.24005	18.14272	18.19505	18.18746	18.12064	18.08482	18.27808	17.79086	17.68666	17.64640
80	24.08079	24.06203	24.10227	24.08717	24.02194	23.98257	24.15624	23.68952	23.57554	23.53352
100	29.88991	29.91633	29.95630	29.94762	29.87145	29.85729	30.00902	29.53279	29.42300	29.38158
120	35.77255	35.84276	35.86089	35.87067	35.80249	35.78976	35.91778	35.44151	35.32785	35.28479
140	41.69945	41.77727	41.81507	41.80701	41.75421	41.73076	41.84973	41.36909	41.25102	41.20509
160	47.58536	47.69247	47.72353	47.72245	47.65645	47.65026	47.77009	47.26797	47.15207	47.10229
180	53.47705	53.57827	53.59913	53.60558	53.55407	53.54149	53.64658	53.14879	53.02676	52.97977
200	59.38878	59.49472	59.52308	59.52575	59.49075	59.47347	59.56541	59.05835	58.92835	58.87296

$\boldsymbol{E}(\tilde{B}/\text{makespan}(G))$, can be considered as a lower bound for $\boldsymbol{E}(\text{efficiency}(G))$, i.e., the *expected energy efficiency* of DSECS-H.

For given n and H, the expected energy efficiency can be obtained experimentally as follows. We generate M random DAGs: $G_1, G_2, ..., G_M$. For each G_k, we run algorithm DSECS-H, get its makespan(G_k) and energy(G_k), calculate the lower bound $\tilde{B}$ with energy budget energy(G_k), and record the ratio $\tilde{B}$/makespan (G_k). The average of the M ratios is returned as a lower bound for the expected energy efficiency $\boldsymbol{E}(\text{efficiency}(G))$. We set $M = 2,000$ for all experiments. The following experimental data demonstrate the expected energy efficiency of algorithm DSECS-H.

In Table 9, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display $\boldsymbol{E}(\tilde{B}/\text{makespan}(G))$ for sparse DAG ($p = 2/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is ±1.35898.

In Table 10, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display $\boldsymbol{E}(\tilde{B}/\text{makespan}(G))$ for sparse DAG ($p = 2/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is ±1.14423%.

In Table 11, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display $\boldsymbol{E}(\tilde{B}/\text{makespan}(G))$ for dense DAG ($p = 5/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is ±1.50151%.

In Table 12, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and H = ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display $\boldsymbol{E}(\tilde{B}/\text{makespan}(G))$ for dense DAG ($p = 5/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is ±1.43717%.

We have the following observations. First, all our heuristic algorithms can achieve reasonably high energy efficiency, especially when n gets large. In particular, for DAG with computation-intensive tasks, the expected energy efficiency can be over 75%; and DAG with communication-intensive tasks, the expected energy efficiency can be over 65%. This means that our heuristic algorithms are able to efficiently utilize all energy resources in processing both computation-intensive and communication-intensive tasks with both sparse and dense precedence constraints on a device-edge-cloud collaborative computing system. Second, the simple ORG strategy seems to perform better than SRF, LRF, SDF, LDF, SRD, and LRD. Furthermore, RAN10, RAN30, and RAN50 can further improve energy efficiency by spending more execution time.

5 Energy-Constrained DAG Scheduling

In this section, we consider energy-constrained DAG scheduling on edge and cloud servers. We define our energy-constrained DAG scheduling problem, present

Table 9 Energy Efficiency of Algorithm DSECS-H for Sparse DAG with Computation-Intensive Tasks ($p = 2/n$, $\gamma = 1.0$, 99% C.I. = ±1.35898%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	0.39761	0.39385	0.39227	0.39723	0.39349	0.39164	0.39812	0.47207	0.48943	0.49532
40	0.58997	0.54196	0.54514	0.54633	0.53804	0.53659	0.54848	0.63778	0.65446	0.66152
60	0.67157	0.61441	0.61800	0.61844	0.60811	0.60212	0.62262	0.70228	0.71500	0.72056
80	0.70760	0.65306	0.65608	0.65488	0.64575	0.63833	0.66025	0.73065	0.74016	0.74431
100	0.73132	0.68000	0.68509	0.68331	0.67136	0.66229	0.68956	0.74904	0.75679	0.76025
120	0.74540	0.69777	0.70361	0.70181	0.68905	0.67964	0.70623	0.76003	0.76712	0.77042
140	0.75479	0.71214	0.71811	0.71743	0.70236	0.69291	0.72061	0.76717	0.77385	0.77651
160	0.76203	0.72344	0.72772	0.72711	0.71214	0.70162	0.73216	0.77307	0.77897	0.78157
180	0.76724	0.73053	0.73722	0.73486	0.72107	0.70933	0.73794	0.77718	0.78263	0.78495
200	0.77189	0.73696	0.74429	0.74184	0.72799	0.71630	0.74404	0.78029	0.78545	0.78772

Table 10 Energy Efficiency of Algorithm DSECS-H for Sparse DAG with Communication-Intensive Tasks ($p = 2/n$, $\gamma = 3.0$, 99% C.I. = ±1.14423%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	0.39593	0.39118	0.38805	0.39366	0.38943	0.39093	0.38940	0.45282	0.46603	0.47070
40	0.53609	0.50460	0.50772	0.50490	0.50871	0.50727	0.50531	0.56965	0.58068	0.58465
60	0.59071	0.55194	0.55425	0.55325	0.55446	0.55426	0.55504	0.60932	0.61676	0.61928
80	0.61206	0.57657	0.58086	0.57772	0.58066	0.57947	0.57992	0.62544	0.63096	0.63333
100	0.62492	0.59504	0.59657	0.59395	0.59738	0.59683	0.59634	0.63518	0.63952	0.64139
120	0.63313	0.60653	0.60796	0.60649	0.60876	0.60965	0.60699	0.64160	0.64541	0.64699
140	0.63899	0.61354	0.61585	0.61476	0.61624	0.61614	0.61677	0.64602	0.64933	0.65070
160	0.64345	0.62043	0.62213	0.62085	0.62359	0.62251	0.62241	0.64948	0.65221	0.65337
180	0.64657	0.62572	0.62722	0.62626	0.62804	0.62712	0.62879	0.65178	0.65444	0.65562
200	0.64926	0.63028	0.63255	0.63082	0.63284	0.63210	0.63307	0.65386	0.65632	0.65735

Table 11 Energy Efficiency of Algorithm DSECS-H for Dense DAG with Computation-Intensive Tasks ($p = 5/n$, $\gamma = 1.0$, 99% C.I. = ±1.50151%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	0.23388	0.25523	0.24218	0.25054	0.24502	0.24900	0.24796	0.29240	0.30246	0.30551
40	0.41593	0.43574	0.42549	0.43280	0.42540	0.42874	0.42842	0.49702	0.51284	0.51896
60	0.55125	0.54851	0.54057	0.54727	0.54290	0.54583	0.54502	0.61613	0.63173	0.63790
80	0.64544	0.61605	0.61458	0.61790	0.61408	0.61331	0.61929	0.68495	0.69762	0.70256
100	0.69472	0.65636	0.65802	0.65862	0.65585	0.65373	0.65781	0.72193	0.73202	0.73607
120	0.72278	0.68263	0.68469	0.68581	0.68060	0.67932	0.68606	0.74366	0.75180	0.75531
140	0.74107	0.70042	0.70278	0.70267	0.69991	0.69857	0.70423	0.75707	0.76387	0.76678
160	0.75149	0.71406	0.71559	0.71535	0.71420	0.70959	0.71705	0.76479	0.77087	0.77379
180	0.75959	0.72386	0.72666	0.72619	0.72322	0.72108	0.72781	0.77160	0.77691	0.77918
200	0.76534	0.73229	0.73335	0.73463	0.73219	0.73016	0.73484	0.77622	0.78117	0.78319

Table 12 Energy Efficiency of Algorithm DSECS-H for Dense DAG with Communication-Intensive Tasks ($p = 5/n$, $\gamma = 3.0$, 99% C.I. = ±1.43717%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	0.24865	0.26649	0.25716	0.26646	0.25808	0.26152	0.26104	0.30549	0.31630	0.31946
40	0.42151	0.43699	0.42524	0.43695	0.42856	0.43270	0.43156	0.48748	0.49967	0.50398
60	0.51919	0.51899	0.51325	0.51912	0.51461	0.51811	0.51535	0.56468	0.57464	0.57773
80	0.57602	0.55986	0.55939	0.56130	0.56060	0.55980	0.56047	0.60157	0.60901	0.61170
100	0.60636	0.58511	0.58403	0.58571	0.58499	0.58440	0.58539	0.62279	0.62793	0.63000
120	0.62108	0.60019	0.59886	0.59873	0.59949	0.59835	0.59843	0.63348	0.63787	0.63960
140	0.63192	0.60951	0.60967	0.61014	0.60952	0.61071	0.61066	0.64127	0.64455	0.64610
160	0.63806	0.61723	0.61812	0.61814	0.61692	0.61750	0.61738	0.64575	0.64891	0.65020
180	0.64208	0.62243	0.62269	0.62273	0.62372	0.62289	0.62218	0.64892	0.65148	0.65261
200	0.64585	0.62690	0.62820	0.62767	0.62856	0.62833	0.62875	0.65172	0.65407	0.65521

our heuristic algorithms, derive a lower bound for the optimal schedule length, and conduct experimental performance evaluation.

5.1 Problem Definition

In this section, we define our energy-constrained DAG scheduling problem.

The most important factors that affect both schedule length and energy consumption are computation and communication speeds. While the computation speeds s_j of the servers and the wired communication speeds w_j cannot be determined by the UE, the computation speed s_0 of the the UE and all wireless communication speeds c_j can be controlled by the UE.

Our energy-constrained DAG scheduling problem can be defined as follows.

Problem 2: Energy-Constrained DAG Scheduling on Edge and Cloud Servers.
Input: Servers $S_0, S_1, S_2, ..., S_m$, a DAG $G = (\mathscr{T}, \prec)$, and energy budget E.
Output: A schedule of G on $S_0, S_1, S_2, ..., S_m$, and the computation speed $s_{0,i}$ or the wireless communication speed $c_{j,i}$ to execute each T_i, where $1 \le i \le n$, such that makespan(G) is minimized and that energy$(G) \le E$.

We assume that E is reasonably large, since there is minimum energy consumption for computation and communication for each task [13].

The above problem is NP-hard even for the following extreme case: (1) tasks are independent, i.e., $\prec = \emptyset$; (2) there is no communication cost, i.e., $d_i = 0$ for all $1 \le i \le n$; (3) there is only one ES, i.e., $m = 1$; (4) $\xi = 1$, $\alpha = 2$, and $P_s = 0$; (5) c_1, b_1, β_1 are unrelated. In this simple case, there is no communication energy consumption but only computation energy consumption on the UE. Hence, we have energy$(G) = (\xi s_0^\alpha + P_s)(R_0/s_0) = E$, that is, $R_0 s_0 = E$, which gives $s_0 = E/R_0$. In the ideal case, the makespan is minimized when S_0 and S_1 have the same execution time, that is, $R_0/s_0 = R_1/s_1$, or, $R_0^2/E = (R - R_0)/s_1$, which yields $s_1 R_0^2 + E R_0 - E R = 0$, and

$$R_0 = \frac{\sqrt{E^2 + 4 s_1 E R} - E}{2 s_1}.$$

The classic *subset sum problem* ([6], p. 223) can be reduced to our problem, in the sense that the set $\{r_1, r_2, ..., r_n\}$ has a subset whose sum is exactly R_0 if and only if makespan$(G) = R_0/s_0$, where R_0 and s_0 are given above. If there is no such a subset, then makespan$(G) > R_0/s_0$.

5.2 The ECDSECS-H Algorithm

In this section, we develop a heuristic algorithm to solve the DAG scheduling problem on edge and cloud servers with energy constraints.

Our energy-constrained DAG scheduling algorithm, called ECDSECS-H (Energy-Constrained DAG Scheduling on Edge and Cloud Servers with Heuristic H, see Algorithm 2), is extended from the DSECS-H algorithm. The ECDSECS-H algorithm chooses an identical computation speed s_0 and an identical wireless communication speed c_j, where $1 \leq j \leq m$, for all tasks.

Algorithm 2: Energy-Constrained DAG Scheduling on Edge and Cloud Servers with Heuristic H (ECDSECS-H)

Input: Servers $S_0, S_1, S_2, ..., S_m$, a DAG $G = (\mathscr{T}, \prec)$, and energy budget E.
Output: A schedule of G on $S_0, S_1, S_2, ..., S_m$, the computation speed s_0, the wireless communication speed c_j, where $1 \leq j \leq m$, such that makespan(G) is minimized and that energy$(G) \leq E$.

---

set $s_0, c_1, c_2, ..., c_m$ to some reasonable initial values; (1)
call algorithm DSECS-H to get makespan(G) and energy(G); (2)
$\phi \leftarrow 0$; (3)
if (energy$(G) > E$) (4)
  repeat //iterative and progressive speed adjustment (5)
    $\phi \leftarrow \phi - \Delta$; (6)
    $s_0 \leftarrow (1+\phi)s_0$; $c_j \leftarrow (1+\phi)c_j$, $1 \leq j \leq m$; //decrease comp/comm speeds (7)
    call algorithm DSECS-H to get makespan(G) and energy(G); (8)
  until (energy$(G) \leq E$); (9)
else if (energy$(G) < E$) (10)
  $\phi \leftarrow \phi + \Delta$; (11)
  $s_0 \leftarrow (1+\phi)s_0$; $c_j \leftarrow (1+\phi)c_j$, $1 \leq j \leq m$; //increase comp/comm speeds (12)
  call algorithm DSECS-H to get makespan$'(G)$ and energy$'(G)$; (13)
  while (energy$'(G) \leq E$) **do** //iterative and progressive speed adjustment (14)
    record the schedule and $s_0, c_1, c_2, ..., c_m$; (15)
    $\phi \leftarrow \phi + \Delta$; (16)
    $s_0 \leftarrow (1+\phi)s_0$; $c_j \leftarrow (1+\phi)c_j$, $1 \leq j \leq m$; //increase comp/comm speeds (17)
    call algorithm DSECS-H to get makespan$'(G)$ and energy$'(G)$; (18)
  end do; (19)
end if. (20)

Initially, s_0 and the c_j's are set to some reasonable initial values (line (1)). Then, algorithm DSECS-H is called as an initial attempt (line (2)). If energy$(G) = E$, the algorithm finishes; otherwise, s_0 and the c_j's are adjusted to satisfy the energy constraint (lines (3)–(20)). The speed of adjustment is controlled by parameters ϕ and Δ. ϕ is initially set to 0 (line (3)) and Δ is fixed.

If energy$(G) > E$ (line (4)), s_0 and the c_j's should be decreased such that energy$(G) \leq E$ (lines (5)–(9)). The value of ϕ, as well as the values of s_0 and the c_j's, are gradually reduced (lines (6)–(7)). Algorithm DSECS-H is called to check the reduced energy(G) (line (8)). The process is repeated until energy$(G) \leq E$ (line (9)).

If energy$(G) < E$ (line (10)), s_0 and the c_j's can be increased while keeping energy$(G) \leq E$ (lines (11)–(19)). The value of ϕ, as well as the values of s_0 and the c_j's, are gradually increased (lines (11)–(12) and (16)–(17)). Algorithm DSECS-H is called to check the increased energy(G) (lines (13) and (18)). An improved schedule is acceptable only if energy$(G) \leq E$ (lines (14–15)).

The initial values of $s_0, c_1, c_2, ..., c_m$ determine the execution time and solution quality of algorithm ECDSECS-H. If these values are far from the optimal values, algorithm ECDSECS-H completes later and produces lower-quality solutions. If these values are close to the optimal values, algorithm ECDSECS-H completes sooner and produces higher-quality solutions.

The value of Δ determines the execution time and solution quality of algorithm ECDSECS-H. If Δ is too big, algorithm ECDSECS-H completes sooner and produces lower-quality solutions. If Δ is too small, algorithm ECDSECS-H completes later and produces higher-quality solutions.

In Section 5.4, we show experimental data of the impact of Δ on the execution time and solution quality of algorithm ECDSECS-H.

5.3 A Lower Bound

In this section, we derive a lower bound for the optimal schedule length in energy-constrained scheduling.

It has been proved in [12] that for the same amount of computation energy consumption, the overall computation time of all tasks executed on the UE is minimized when all tasks have the same computation speed s_0. Hence, the total computation energy consumption on the UE is $((\xi s_0^{\alpha} + P_s)/s_0)R_0$.

Furthermore, it has also been proved in [12] that for the same amount of communication energy consumption, the overall wireless communication time of all tasks executed on S_j is minimized when all tasks have the same wireless communication speed c_j. Hence, the total communication energy consumption for S_j is $((2^{c_j/b_j} - 1)/(\beta_j c_j))D_j = \gamma((2^{c_j/b_j} - 1)/(\beta_j c_j))R_j$.

We consider the same B defined in Section 3.3, i.e.,

$$B = \max\{R_0/s_0, R_1/s_1, R_2/s_2, ..., R_m/s_m, D_1/c_1 + D_2/c_2 + \cdots + D_m/c_m\}.$$

We need to minimize B, subject to the condition $R_0 + R_1 + R_2 + \cdots + R_m = R$, and

$$((\xi s_0^\alpha + P_s)/s_0)R_0 + \sum_{j=1}^{m} \gamma((2^{c_j/b_j} - 1)/(\beta_j c_j))R_j = E,$$

by choosing s_0 and $c_1, c_2, ..., c_m$. The minimized B can be a lower bound for the optimal schedule length.

Again, we consider $R_j = (s_j/S)R$, for all $0 \le j \le m$, such that B is minimized. The remaining issue is how to minimize the wireless communication time, i.e., $D_1/c_1 + D_2/c_2 + \cdots + D_m/c_m$.

The total wireless communication time is

$$T' = \sum_{j=1}^{m} D_j/c_j = \sum_{j=1}^{m} \gamma(R_j/c_j)$$
$$= \sum_{j=1}^{m} \gamma(R/S)(s_j/c_j) = \gamma(R/S)\sum_{j=1}^{m} s_j/c_j.$$

The total wireless communication energy consumption is

$$E' = \sum_{j=1}^{m} \gamma(2^{c_j/b_j} - 1)/(\beta_j c_j)R_j$$
$$= \sum_{j=1}^{m} \gamma(R/S)s_j(2^{c_j/b_j} - 1)/(\beta_j c_j)$$
$$= \gamma(R/S)\sum_{j=1}^{m} s_j(2^{c_j/b_j} - 1)/(\beta_j c_j).$$

Both T' and E' can be treated as functions of $c_1, c_2, ..., c_m$, i.e., $T'(c_1, c_2, ..., c_m)$ and $E'(c_1, c_2, ..., c_m)$.

For a given s_0, the total computation energy consumption on the UE is

$$((\xi s_0^\alpha + P_s)/s_0)R_0 = (\xi s_0^\alpha + P_s)(R/S).$$

We try to minimize $T'(c_1, c_2, ..., c_m)$ subject to the condition

$$E'(c_1, c_2, ..., c_m) = E - (\xi s_0^\alpha + P_s)(R/S).$$

To this end, we use the Lagrange multiplier system:

$$\nabla T'(c_1, c_2, ..., c_m) = \lambda \nabla E'(c_1, c_2, ..., c_m),$$

where λ is a Lagrange multiplier. The above equation is actually

$$\partial T'(c_1, c_2, ..., c_m)/\partial c_j = \lambda \partial E'(c_1, c_2, ..., c_m)/\partial c_j,$$

that is,

$$-\frac{s_j}{c_j^2} = \lambda s_j \cdot \frac{2^{c_j/b_j}(\ln 2/b_j)c_j - (2^{c_j/b_j} - 1)}{\beta_j c_j^2},$$

and equivalently,

$$\frac{2^{c_j/b_j}((\ln 2/b_j)c_j - 1) + 1}{\beta_j} = -\frac{1}{\lambda},$$

for all $1 \le j \le m$.

Our numerical procedure to find s_0 and $c_1, c_2, ..., c_m$ is as follows.

First, for a given λ, we can find c_j numerically by noticing that the left-hand side of the above equation is an increasing function of c_j.

Second, by choosing λ appropriately, we can decide $c_1, c_2, ..., c_m$ such that $E'(c_1, c_2, ..., c_m) = E - (\xi s_0^\alpha + P_s)(R/S)$ by noticing that $E'(c_1, c_2, ..., c_m)$ is an increasing function of λ.

Finally, by choosing s_0 appropriately, we can decide s_0 such that $T'(c_1, c_2, ..., c_m) = R/S$ by noticing that $T'(c_1, c_2, ..., c_m) - R/S$ is an increasing function of s_0.

The above discussion is summarized in the following theorem.

Theorem 2 *A lower bound for the optimal makespan in energy-constrained DAG scheduling is*

$$\text{makespan}^*(G) \ge R/S = R/(s_0 + s_1 + s_2 + \cdots + s_m),$$

where s_0 is obtained using the above numerical procedure for the energy budget E.

We use $\tilde{B}$ to denote the lower bound in Theorem 2.

5.4 Performance Evaluation

In this section, we experimentally evaluate the performance of algorithm ECDSECS-H.

We keep the same parameter settings of Sections 3.4 and 4.2. The energy budget is $E = 0.235n$ for a DAG with n computation-intensive tasks and $E = 0.3n$

for a DAG with n communication-intensive tasks. The parameter Δ is 0.01. (Notice that the values of our energy budget E are taken from Tables 5–8. However, observations in this section should still hold for other values of E.)

5.4.1 Expected Performance Bound

For given n and H, the expected performance bound can be obtained experimentally as follows. We generate M random DAGs: $G_1, G_2, ..., G_M$. For each G_k, we run algorithm ECDSECS-H with energy budget E, get its makespan(G_k), calculate the lower bound $\tilde{B}$ with energy constraint E, and record the ratio makespan$(G_k)/\tilde{B}$. The average of the M ratios is returned as the expected performance bound. We set $M = 2,000$ for all experiments.

The following experimental data demonstrate the expected performance bound of algorithm ECDSECS-H.

In Table 13, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for sparse DAG ($p = 2/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is $\pm 2.71486\%$.

In Table 14, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for sparse DAG ($p = 2/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 2.13717\%$.

In Table 15, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for dense DAG ($p = 5/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is $\pm 2.75421\%$.

In Table 16, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $H =$ ORG, SRF, LRF, SDF, LDF, SRD, LRD, RAN10, RAN30, RAN50, we display the expected performance bound for dense DAG ($p = 5/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 2.66240\%$.

We have the following observations.

First, all our heuristic algorithms can produce near-optimal schedules whose makespans are reasonably close to the optimal makespans, in the sense that the expected performance bounds are reasonably close to one, especially when n gets large. This means that our heuristic algorithms are able to efficiently utilize all communication, computation, and energy resources in processing both computation-intensive and communication-intensive tasks with both sparse and dense precedence constraints on a device-edge-cloud collaborative computing system.

Second, the simple ORG strategy seems to perform better than SRF, LRF, SDF, LDF, SRD, and LRD, except for small n and dense DAG. Furthermore, RAN10, RAN30, and RAN50 can further improve the scheduling performance by spending more execution time.

Table 13 Expected Performance Bound of Algorithm ECDSECS-H for Sparse DAG with Computation-Intensive Tasks ($p = 2/n$, $\gamma = 1.0$, 99% C.I. = $\pm 2.71486\%$)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	3.18607	3.00364	3.18489	3.08685	3.03324	2.98346	3.18168	2.24781	2.20330	2.18627
40	1.71462	1.91488	1.90237	1.91783	1.91166	1.91393	1.90850	1.58641	1.55911	1.54772
60	1.48682	1.66184	1.64178	1.65335	1.65748	1.66878	1.63900	1.43152	1.41378	1.40664
80	1.38900	1.53474	1.51982	1.51970	1.53996	1.55733	1.51123	1.35722	1.34400	1.33901
100	1.34503	1.45976	1.45032	1.44487	1.47574	1.49012	1.43936	1.32004	1.30896	1.30394
120	1.31881	1.41816	1.40713	1.40798	1.43303	1.45314	1.40121	1.29884	1.28907	1.28446
140	1.29781	1.38292	1.37658	1.37446	1.40428	1.42227	1.36872	1.28122	1.27261	1.26876
160	1.28771	1.36228	1.35917	1.35429	1.38641	1.40552	1.35216	1.27314	1.26498	1.26105
180	1.27263	1.33873	1.33371	1.33099	1.36770	1.39049	1.32678	1.25839	1.25084	1.24719
200	1.26671	1.33073	1.32279	1.32200	1.35705	1.37685	1.31737	1.25297	1.24554	1.24236

Table 14 Expected Performance Bound of Algorithm ECDSECS-H for Sparse DAG with Communication-Intensive Tasks ($p = 2/n$, $\gamma = 3.0$, 99% C.I. = ±2.13717%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	2.92931	2.89041	3.00328	2.88611	2.93881	2.86721	2.95327	2.30313	2.25512	2.23828
40	1.91126	2.06605	2.05666	2.06842	2.04653	2.04172	2.06559	1.77593	1.74707	1.73607
60	1.69692	1.83956	1.83075	1.83773	1.81980	1.81845	1.83285	1.63667	1.62009	1.61231
80	1.62163	1.73402	1.71703	1.73502	1.70995	1.71488	1.72976	1.58228	1.56828	1.56266
100	1.58181	1.67225	1.66511	1.67377	1.65603	1.65859	1.66579	1.55211	1.54116	1.53613
120	1.56008	1.64074	1.63244	1.64185	1.62638	1.62630	1.63437	1.53667	1.52724	1.52279
140	1.54407	1.61534	1.60524	1.61332	1.60105	1.60089	1.60743	1.52439	1.51631	1.51222
160	1.53529	1.59470	1.58507	1.59351	1.58351	1.58231	1.58774	1.51693	1.50846	1.50487
180	1.52378	1.57867	1.57437	1.57802	1.57315	1.57125	1.57204	1.50854	1.50122	1.49802
200	1.52214	1.57342	1.56620	1.57443	1.56215	1.56300	1.56772	1.50725	1.49990	1.49655

Table 15 Expected Performance Bound of Algorithm ECDSECS-H for Dense DAG with Computation-Intensive Tasks ($p = 5/n$, $\gamma = 1.0$, 99% C.I. = ±2.75421%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	7.45765	5.64024	6.78798	6.11835	6.39495	6.04709	6.44084	3.97143	3.88615	3.86401
40	2.88418	2.54009	2.73190	2.59724	2.64794	2.57632	2.64450	2.08832	2.04252	2.02565
60	1.84088	1.87222	1.90087	1.89440	1.89559	1.87027	1.88425	1.64223	1.61137	1.59935
80	1.52267	1.62310	1.62640	1.63174	1.62895	1.62164	1.62423	1.45800	1.43853	1.43029
100	1.41208	1.51648	1.51670	1.51996	1.52405	1.52761	1.50812	1.37702	1.36394	1.35810
120	1.35206	1.44787	1.44533	1.44869	1.45279	1.45298	1.44633	1.32688	1.31687	1.31214
140	1.31868	1.40416	1.40019	1.40284	1.40600	1.41110	1.39899	1.29907	1.28963	1.28530
160	1.29923	1.37591	1.37320	1.37452	1.38017	1.38627	1.36914	1.28209	1.27388	1.27019
180	1.28824	1.35750	1.35206	1.35553	1.35855	1.36321	1.35185	1.27123	1.26355	1.26004
200	1.27097	1.33458	1.33085	1.32985	1.33765	1.34306	1.32544	1.25636	1.24912	1.24557

Table 16 Expected Performance Bound of Algorithm ECDSECS-H for Dense DAG with Communication-Intensive Tasks ($p = 5/n$, $\gamma = 3.0$, 99% C.I. = ±2.66240%)

n	ORG	SRF	LRF	SDF	LDF	SRD	LRD	RAN10	RAN30	RAN50
20	5.89939	4.69521	5.43295	4.72985	5.30305	4.97625	5.10744	3.54697	3.47463	3.45219
40	2.62849	2.43332	2.55528	2.44687	2.53154	2.47490	2.50718	2.10484	2.06394	2.04921
60	1.94340	1.96079	1.98190	1.95637	1.96803	1.95900	1.97596	1.77271	1.74643	1.73615
80	1.73367	1.79212	1.78469	1.79257	1.78194	1.77850	1.79203	1.65005	1.63192	1.62445
100	1.64241	1.70551	1.70271	1.70957	1.70156	1.69993	1.70661	1.59330	1.57839	1.57246
120	1.59596	1.65916	1.65737	1.66169	1.65737	1.65316	1.65944	1.56221	1.55025	1.54482
140	1.56473	1.62377	1.61829	1.62244	1.61345	1.61219	1.62191	1.53694	1.52733	1.52335
160	1.55227	1.60507	1.60337	1.60608	1.60007	1.59823	1.60329	1.52876	1.51969	1.51576
180	1.53688	1.58468	1.58193	1.58573	1.57794	1.58004	1.58407	1.51578	1.50771	1.50414
200	1.52685	1.57206	1.56897	1.57253	1.56652	1.56632	1.57219	1.50889	1.50161	1.49844

5.4.2 Impact of Δ

For given n and Δ, we generate M random DAGs: $G_1, G_2, ..., G_M$. For each G_k, we run algorithm ECDSECS-H with H = ORG, get its makespan(G_k) and the number C_k of times that algorithm DSECS-H is called, calculate the lower bound $\tilde{B}$ with energy constraint E, and record the ratio makespan$(G_k)/\tilde{B}$. The average of the M ratios and the average of $C_1, C_2, ..., C_M$ are reported. We set $M = 10,000$ for all experiments.

The following experimental data demonstrate the impact of Δ on the execution time and solution quality of algorithm ECDSECS-H.

In Table 17, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $\Delta = 0.01, 0.02, 0.03, 0.04, 0.05$, we display the expected performance bound and the expected number of times that algorithm DSECS-H is called, for sparse DAG ($p = 2/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is $\pm 2.09\%$.

In Table 18, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $\Delta = 0.01, 0.02, 0.03, 0.04, 0.05$, we display the expected performance bound and the expected number of times that algorithm DSECS-H is called, for sparse DAG ($p = 2/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 2.05\%$.

In Table 19, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $\Delta = 0.01, 0.02, 0.03, 0.04, 0.05$, we display the expected performance bound and the expected number of times that algorithm DSECS-H is called, for dense DAG ($p = 5/n$) with computation-intensive tasks ($\gamma = 1.0$), where the 99% confidence interval (C.I.) is $\pm 1.88\%$.

In Table 20, for each combination of $n = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200$, and $\Delta = 0.01, 0.02, 0.03, 0.04, 0.05$, we display the expected performance bound and the expected number of times that algorithm DSECS-H is called, for dense DAG ($p = 5/n$) with communication-intensive tasks ($\gamma = 3.0$), where the 99% confidence interval (C.I.) is $\pm 1.76\%$.

We have the following observations. First, as Δ becomes small, the expected performance bound decreases; that is, algorithm ECDSECS-H produces better schedules. Second, as Δ becomes small, algorithm DSECS-H is called more times and algorithm ECDSECS-H takes longer time. However, the expected number of times that algorithm DSECS-H is called is quite small, unless when n is very small.

6 Related Research

In this section, we review related research.

DAG scheduling in mobile edge computing has been investigated by several researchers with diversified optimization objectives such as execution latency minimization, energy consumption minimization, and execution reliability maximization. Cai et al. developed a context-aware greedy task scheduling algorithm to minimize the task completion time and a dependency-aware task rescheduling algorithm to cope with the failure of edge servers [2]. Duan et al. proposed a resource pipeline scheme with the objective of minimizing the makespan in DAG scheduling [4]. Li et

Table 17 Impact of Δ on Algorithm ECDSECS-H for Sparse DAG with Computation-Intensive Tasks ($p = 2/n$, $\gamma = 1.0$, 99% C.I. = $\pm 2.09\%$)

n	$\Delta = 0.01$	$\Delta = 0.02$	$\Delta = 0.03$	$\Delta = 0.04$	$\Delta = 0.05$
20	3.27190, 21.04	3.28772, 12.00	3.31245, 8.90	3.33983, 7.30	3.38677, 6.30
40	1.76322, 10.16	1.77165, 6.95	1.78017, 5.69	1.79144, 4.96	1.79974, 4.45
60	1.50107, 8.28	1.50800, 6.26	1.51594, 5.35	1.52310, 4.71	1.53159, 4.24
80	1.40684, 7.53	1.41261, 5.92	1.41979, 5.11	1.42605, 4.52	1.43409, 4.05
100	1.35569, 7.22	1.36168, 5.84	1.36791, 5.07	1.37416, 4.48	1.38011, 4.00
120	1.32667, 6.87	1.33182, 5.64	1.33729, 4.89	1.34268, 4.32	1.34908, 3.85
140	1.30699, 6.70	1.31225, 5.54	1.31711, 4.81	1.32274, 4.23	1.32909, 3.76
160	1.29097, 6.51	1.29562, 5.43	1.30035, 4.71	1.30579, 4.14	1.31195, 3.68
180	1.27899, 6.41	1.28307, 5.37	1.28735, 4.66	1.29282, 4.09	1.29882, 3.61
200	1.27026, 6.34	1.27412, 5.33	1.27873, 4.62	1.28377, 4.05	1.29036, 3.57

Table 18 Impact of Δ on Algorithm ECDSECS-H for Sparse DAG with Communication-Intensive Tasks ($p = 2/n$, $\gamma = 3.0$, 99% C.I. = ±2.05%)

n	$\Delta = 0.01$	$\Delta = 0.02$	$\Delta = 0.03$	$\Delta = 0.04$	$\Delta = 0.05$
20	2.94183, 14.66	2.95639, 9.03	2.96934, 7.00	2.98714, 5.90	3.00443, 5.19
40	1.90698, 8.61	1.91422, 6.17	1.92284, 5.14	1.93057, 4.50	1.94071, 4.03
60	1.70086, 7.13	1.70721, 5.43	1.71419, 4.61	1.72188, 4.05	1.72970, 3.62
80	1.62634, 6.42	1.63231, 4.99	1.63940, 4.25	1.64702, 3.73	1.65649, 3.33
100	1.58766, 6.03	1.59294, 4.75	1.59959, 4.04	1.60671, 3.54	1.61542, 3.15
120	1.56363, 5.67	1.56906, 4.50	1.57578, 3.84	1.58299, 3.36	1.59171, 2.99
140	1.54690, 5.48	1.55203, 4.38	1.55858, 3.73	1.56622, 3.26	1.57461, 2.90
160	1.53657, 5.29	1.54159, 4.24	1.54820, 3.60	1.55563, 3.14	1.56447, 2.79
180	1.52751, 5.13	1.53247, 4.11	1.53893, 3.49	1.54698, 3.04	1.55602, 2.71
200	1.52127, 4.97	1.52649, 4.00	1.53264, 3.39	1.54074, 2.95	1.54916, 2.63

Table 19 Impact of Δ on Algorithm ECDSECS-H for Dense DAG with Computation-Intensive Tasks ($p = 5/n$, $\gamma = 1.0$, 99% C.I. = ±1.88%)

n	$\Delta = 0.01$	$\Delta = 0.02$	$\Delta = 0.03$	$\Delta = 0.04$	$\Delta = 0.05$
20	7.40213, 39.22	7.42118, 20.30	7.53338, 14.15	7.64914, 11.06	7.97806, 9.39
40	2.90243, 17.03	2.91927, 9.61	2.94303, 7.09	2.96399, 5.78	2.98735, 4.98
60	1.86436, 9.24	1.87742, 6.05	1.89201, 4.86	1.90475, 4.19	1.92048, 3.74
80	1.55042, 7.26	1.56194, 5.37	1.57150, 4.54	1.58262, 3.99	1.59255, 3.58
100	1.42973, 6.88	1.43866, 5.35	1.44587, 4.59	1.45525, 4.04	1.46281, 3.61
120	1.36725, 6.66	1.37449, 5.36	1.38049, 4.63	1.38716, 4.09	1.39548, 3.65
140	1.32997, 6.51	1.33558, 5.34	1.34130, 4.63	1.34748, 4.08	1.35517, 3.63
160	1.30861, 6.42	1.31396, 5.31	1.31860, 4.62	1.32433, 4.06	1.33133, 3.60
180	1.29179, 6.35	1.29636, 5.29	1.30145, 4.59	1.30684, 4.03	1.31327, 3.56
200	1.28084, 6.20	1.28521, 5.20	1.28960, 4.50	1.29522, 3.94	1.30125, 3.47

Table 20 Impact of Δ on Algorithm ECDSECS-H for Dense DAG with Communication-Intensive Tasks ($p = 5/n$, $\gamma = 3.0$, 99% C.I. = ±1.76%)

n	$\Delta = 0.01$	$\Delta = 0.02$	$\Delta = 0.03$	$\Delta = 0.04$	$\Delta = 0.05$
20	6.02406, 28.02	6.04898, 14.90	6.11113, 10.55	6.17086, 8.35	6.31649, 7.09
40	2.62300, 10.88	2.63486, 6.77	2.64656, 5.27	2.65957, 4.45	2.67444, 3.91
60	1.96011, 7.47	1.96869, 5.25	1.97955, 4.34	1.99037, 3.76	2.00251, 3.36
80	1.74361, 6.37	1.75094, 4.74	1.76001, 3.99	1.76852, 3.49	1.78026, 3.13
100	1.64477, 5.93	1.65196, 4.57	1.65954, 3.88	1.66849, 3.39	1.67828, 3.03
120	1.59735, 5.62	1.60362, 4.41	1.61144, 3.73	1.61933, 3.26	1.62914, 2.91
140	1.57024, 5.32	1.57613, 4.21	1.58360, 3.57	1.59218, 3.12	1.60038, 2.78
160	1.55001, 5.25	1.55568, 4.19	1.56253, 3.55	1.57038, 3.10	1.57908, 2.76
180	1.53796, 5.11	1.54323, 4.09	1.55011, 3.47	1.55799, 3.02	1.56693, 2.69
200	1.52911, 4.90	1.53451, 3.93	1.54134, 3.34	1.54940, 2.90	1.55840, 2.59

al. studied online placing and scheduling dependent tasks with deadlines and on-demand function configuration on edge servers, aiming to satisfy as many deadlines as possible [11]. Li et al. employed deep reinforcement learning to develop DAG task scheduling algorithms and UAV (unmanned aerial vehicles) deployment optimization algorithms [12]. Li developed a class of pre-power-allocation algorithms and a class of post-power-allocation algorithms for both energy-constrained and time-constrained precedence constrained tasks scheduling [15]. Li et al. proposed an orchestration framework to reduce both execution latency and failure probability for applications having interdependent tasks [18]. Liang et al. tried to minimize makespan in DAG scheduling through offloading order adjusting and execution frequency scaling [19]. Liu et al. formulated DAG task scheduling as an integer programming problem to minimize the overall task completion time while ensuring a high execution success rate [20]. Shang et al. considered task scheduling for DAG-based applications in MEC to maximize the execution reliability given energy consumption and execution latency constraints [22]. Zhu et al. proposed a multi-objective cuckoo search algorithm to minimize execution latency and energy consumption with execution reliability constraint [27].

Several researchers have studied task scheduling in device-edge-cloud collaborative computing environments. Dreibholz and Mazumdar proposed a lightweight task-scheduling framework from a cloud service provider perspective, for applications using both cloud and edge platforms [3]. Li designed heuristic algorithms for scheduling independent tasks on multiple cloud-assisted edge servers with energy constraints [17]. Ma et al. solved the dynamic task scheduling problem in cloud-assisted mobile edge computing (including both peer task scheduling among edge nodes and cross-layer task scheduling from edge nodes to the cloud), aiming at minimizing average task response time within resource budget limit [21]. Yin et al. developed a multi-objective task scheduling mechanism and integrated it into a cloud-edge computing framework for intelligent production lines, aiming to reduce service latency and energy consumption by using particle swarm optimization and gravitational search algorithm [25]. Zhang et al. established a hierarchical architecture for edge-cloud collaborative environments to reduce delay and latency in dynamic real-time task scheduling with deadlines and time sensitivity [26]. However, there is a lack of DAG scheduling in device-edge-cloud collaborative computing environments, and this paper has made efforts in this direction.

Energy-aware DAG and workflow scheduling in data center networks, distributed cloud, edge-cloud, and cloud-edge environments has also been considered by some researchers. Fraga et al. proposed a heuristic approach to scheduling real-time flows in data center networks to optimize the temporal requirements while reducing the energy consumption in the network infrastructure [5]. Hussain et al. developed a deadline-constrained energy-aware workflow scheduling algorithm to minimize energy costs when scheduling workflow tasks on heterogeneous servers in geographically distributed cloud data centers [8]. Jayanetti et al. adopted a deep reinforcement learning model to establish a desired trade-off between the conflicting objectives of energy optimization and time minimization for precedence-constrained tasks scheduling in edge-cloud environments [10]. Wen and Xu designed an improved genetic algorithm to optimize energy consumption under time delay constraints in task offloading scheduling [23]. Xiao et al. proposed a heterogeneous earliest completion time algorithm for workflow scheduling by considering computing performance, transmission delay, energy consumption, and cost of cloud and edge nodes [24].

Energy-constrained task scheduling on heterogeneous edge and cloud servers in the context of combinatorial optimization has been addressed before by the author [13,15–17]. However, among the major features mentioned in this paper, while all these papers studied energy-constrained scheduling by speed setting, only [16,17] considered edge servers and cloud servers (i.e., wireless and wired communications), only [15] considered precedence constraint, only [16,17] supported sequential wireless communications, none implemented overlapped communication and computation, and only [16,17] derived lower bounds for optimal makespan. In this paper, we simultaneously included and incorporated edge and cloud collaborative computing, precedence constraint, sequential transmission, overlapped communication and computation, comparison with optimal solutions, energy budget, computation and communication speed setting in heuristic DAG-scheduling on multiple heterogeneous edge and cloud servers.

Feature	[13]	[15]	[16]	[17]	this paper
edge and cloud fusion	×	×	✓	✓	✓
precedence constraint	×	✓	×	×	✓
sequential transmission	×	×	✓	✓	✓
overlapped communication and computation	×	×	×	×	✓
optimal makespan	×	×	✓	✓	✓
energy constraint	✓	✓	✓	✓	✓
computation and communication speed setting	✓	✓	✓	✓	✓

It is noticed that in all the above work, there is little existing paper that includes performance comparison with optimal solutions, except [16, 17].

7 Summary

We have studied the NP-hard problems of DAG scheduling and energy-constrained DAG scheduling on mobile devices, edge servers, and cloud servers, where all wireless communications are performed sequentially and wireless communication of one task can overlap with wired communication and computation of another task on the same server. We have designed and evaluated new heuristic algorithms. One strong and unique feature of our study is to derive a lower bound for the optimal makespan such that the solutions of our heuristic algorithms can be compared with optimal solutions. These efforts and investigations have not been seen in the existing literature. This paper has made significant and substantial contributions to DAG scheduling in device-edge-cloud collaborative computing systems.

Further research can be conducted towards the following directions. First, more effective and efficient heuristic algorithms with better performance (in terms of both solution quality and execution time) should be designed and tighter lower bounds should be derived. Second, joint optimization of schedule length and energy consumption, together with the performance-cost trade-off, can be studied. Third, more sophisticated application environments with multiple mobile devices and user equipments sharing and competing for edge and cloud servers can be considered.

Appendix: Notations and Definitions

Notation	Definition
m_1	the number of edge servers (ES)
m_2	the number of cloud servers (CS)
m	$= m_1 + m_2$, the number of servers
S_0	the UE
S_0'	a virtual server responsible for all wireless communications of S_0
S_j	the jth server (either an ES or a CS), $1 \le j \le m$
ES_j	S_j if S_j is an ES
CS_j	S_j if S_j is a CS
CF_j	the communication frontend of CS_j
s_j	the computation speed (measured by Bips) of S_j
c_j	the wireless communication speed (measured by Mbps) of S_j
w_j	the wired communication speed (measured by Mbps) of S_j (if S_j is a CS)
G	$= (\mathscr{T}, \prec)$, a directed acyclic graph
$\mathscr{T}$	$= \{T_1, T_2, ..., T_n\}$, a set of tasks
$\prec$	$\subseteq \mathscr{T} \times \mathscr{T}$, a set of precedence constraints
n	the number of tasks
T_i	$= (d_i, r_i)$, the ith task, $1 \le i \le n$
d_i	the amount of communication (measured by MB) of T_i
r_i	the amount of computation (measured by BI) of T_i
t_i	the execution time of T_i
H	a heuristic
CompRT_j	the computation ready time of S_j, $0 \le j \le m$
CommRT	the wireless communication ready time of S_0'
clock	a global clock
makespan(G)	$= \max_{0 \le j \le m}\{\mathrm{CompRT}_j\}$, the makespan of G
makespan$^*(G)$	the optimal schedule length of G
γ	$= \min_{1 \le i \le n}\{d_i / r_i\}$
R_j	the total amount of computation on S_j
D_j	the total amount of communication of tasks processed on S_j
W_j	the total waiting time of S_j
R	$= R_0 + R_1 + R_2 + \cdots + R_m$, the total amount of computation
S	$= s_0 + s_1 + s_2 + \cdots + s_m$, the aggregated computation speed

Notation	Definition
B	a lower bound for the optimal schedule length
P_0	computation power consumption of the UE
P_s	static power consumption of the UE
ξ, α	parameters of the computation power consumption model
P_j	power consumption of the wireless communication channel between the UE and S_j
b_j	the channel bandwidth
β_j	a combined quantity of several factors of a wireless communication channel
E_i	the energy consumption of T_i
energy(G)	the total energy consumption of G
ϕ, Δ	control parameters of the ECDSECS-H algorithm

Acknowledgements The author would like to express his gratitude to the anonymous reviewers for their careful review and useful suggestions for improving the manuscript.

Author contributions This is a single-authored paper.

Data Availability Statement No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Avan, A., Azim, A., Mahmoud, Q.H.: A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective. Electronics **12**(12), 2599 (2023)
2. Cai, L., Wei, X., Xing, C., Zou, X., Zhang, G., Wang, X.: Failure-resilient DAG task scheduling in edge computing. Comput. Netw. **198**, 108361 (2021)
3. Dreibholz, T., Mazumdar, S.: Towards a lightweight task scheduling framework for cloud and edge platform. Int. Things **21**, 100651 (2023)
4. Duan, Y., Wang, N., Wu, J.: Accelerating DAG-style job execution via optimizing resource pipeline scheduling. J Comput Sci Tech **37**, 852–868 (2022)
5. Fraga, M., Micheletto, M., Llinás, A., Santos, R., Zabala, P.: Flow scheduling in data center networks with time and energy constraints: A software-defined network approach. Future Int. **14**(2), 65 (2022)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability - A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
7. Graham, R.L.: Bounds on multiprocessing timing anomalies SIAM. J. Appl. Math. **2**, 416–429 (1969)
8. Hussain, M., Wei, L.-F., Rehman, A., Abbas, F., Hussain, A., Ali, M.: Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. Future Gener Comput. Syst. **132**, 211–222 (2022)
9. Jamil, B., Ijaz, H., Shojafar, M., Munir, K., Buyya, R.: Resource allocation and task scheduling in fog computing and Internet of Everything environments: A taxonomy, review, and future directions. ACM Comput. Surv. **54**(11s), Article No. 233, 1–38 (2022)
10. Jayanetti, A., Halgamuge, S., Buyya, R.: Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge-cloud computing environments. Future Gener. Comput. Syst. **137**, 14–30 (2022)
11. Li, G., Tan, H., Liu, L., Zhou, H., Jiang, S.H.-C., Han, Z., Li, X.-Y., Chen, G.: DAG scheduling in mobile edge computing. ACM Trans. Sensor Netw. **20**(1) Article No. 12, 1–25 (2023)
12. Li, J., Pan, Y., Xia, Y., Fan, Z., Wang, X., Lv, J.: Optimizing dag scheduling and deployment for IoT data analysis services in the multi-UAV mobile edge computing system. Wireless Networks, published online (2023)
13. Li, K.: Heuristic computation offloading algorithms for mobile users in fog computing. ACM Trans. Embedded Comput. Syst. **20**(2) Article 11, 28 (2021)
14. Li, K.: Distributed and individualized computation offloading optimization in a fog computing environment. J. Parallel Distrib. Comput. **159**, 24–34 (2022)
15. Li, K.: Scheduling precedence constrained tasks for mobile applications in fog computing. IEEE Trans. Serv. Comput. **16**(3), 2153–2164 (2023)
16. Li, K.: Design and analysis of heuristic algorithms for energy-constrained task scheduling with device-edge-cloud fusion. IEEE Trans. Sustain. Comput. **8**(2), 208–221 (2023)
17. Li, K.: Scheduling independent tasks on multiple cloud-assisted edge servers with energy constraint. J. Parallel Distrib. Comput. **184**, 104781 (2024)
18. Li, X., Abdallah, M., Suryavansh, S., Chiang, M., Kim, K.T., Bagchi, S.: DAG-based task orchestration for edge computing. 41st International Symposium on Reliable Distributed Systems, Vienna, Austria (2022)
19. Liang, J., Li, K., Liu, C., Li, K.: Joint offloading and scheduling decisions for DAG applications in mobile edge computing. Neurocomputing **424**, 160–171 (2021)
20. Liu, Z., Liwang, M., Hosseinalipour, S., Dai, H., Gao, Z., Huang, L.: RFID: Towards low latency and reliable DAG task scheduling over dynamic vehicular clouds. IEEE Trans. Vehicul. Tech. **72**(9), 12139–12153 (2023)
21. Ma, X., Zhou, A., Zhang, S., Li, Q., Liu, A.X., Wang, S.: Dynamic task scheduling in cloud-assisted mobile edge computing. IEEE Trans. Mobile Comput. **22**, 2116–2130 (2023)
22. Shang, Y., Li, J., Wu, X.: DAG-based task scheduling in mobile edge computing. 7th International Conference on Information Science and Control Engineering, Changsha, China (2020)
23. Wen, S., Xu, H.: Task offloading scheduling with time constraint for optimizing energy consumption in edge cloud computing. Open Access Library J. **10**, e10910 (2023)
24. Xiao, H., Yuan, J., Wang, N.: A budget-constrained energy-efficient scheduling algorithm on cloud-edge collaborative

workflows. IEEE 25th International Conference on Computer Supported Cooperative Work in Design, Hangzhou, China (2022)
25. Yin, H., Huang, X., Cao, E.: A cloud-edge-based multi-objective task scheduling approach for smart manufacturing lines. J. Grid Comput. **22**, 9 (2024)
26. Zhang, Y., Tang, B., Luo, J., Zhang, J.: Deadline-aware dynamic task scheduling in edge-cloud collaborative computing. Electronics **11**(15), 2464 (2022)
27. Zhu, L., Shang, Y., Li, J., Jia, Y., Yang, Q.: Reliability-constrained task scheduling for DAG applications in mobile edge computing. Wirel. Commun. Mobile Comput. **2024**, 6980514, 12 (2024)