Enabling Streaming Analytics in Satellite Edge Computing via Timely Evaluation of Big Data Queries

Zichuan Xu, Member, IEEE, Guangyuan Xu, Hao Wang, Member, IEEE, Weifa Liang, Senior Member, IEEE, Qiufen Xia, Member, IEEE, and Shangguang Wang, Senior Member, IEEE

Abstract—Internet-of-Things (IoT) applications from many industries, such as transportation (maritime, road, rail, air) and fleet management, offshore monitoring, and farming are located in remote areas without cellular connectivity. Such IoT applications continuously generate stream data with hidden values that need to be unveiled in real time. Streaming analytics is emerging as a popular type of big data analytics to process large volume of stream data for IoT applications in remote regions. Built upon terrestrial-satellite integrated networks, Satellite Edge Computing (SEC) equipped with computing resource in satellites has been envisioning as a key enabling technology to timely analyze stream data of IoT applications in remote regions on the Earth. Considering the dynamically-moving property of satellites in an SEC network, it is vital to optimize the responsiveness of each big data analytical query, such that none of such queries takes much longer time to wait for available satellites with sufficient computing resource. Furthermore, the uncertain data volumes of big data queries in SEC networks make the resources in satellites fragmented, particularly when the collaboration among satellites is intermittent. Therefore, directly application of existing methods may not guarantee the timeliness of streaming analytics in an SEC network.

To address the aforesaid unique challenges of streaming analytics in SEC networks, it is urgent to design new algorithms and methods for timely big data processing. Specifically, we use the flow time to capture the responsiveness of streaming analytics in satellite edge computing, which is the time between the generation of the first unit of a dataset and the finish time of the data processing. We consider the flow time minimization problem for query evaluation of big data analytics in an SEC network with the aim of minimizing the average flow time of big data analytical queries, under an assumption of uncertain volumes of datasets. To this end, we first propose an approximation algorithm with a provable approximation ratio for the offline version of the flow time minimization problem. We then devise an online learning algorithm, referred to the customized Lipschitz bandit learning algorithm, with a bounded regret for the online version of the problem. We finally evaluate the performance of the proposed algorithms in a real SEC network topology. Experiment results show that the performance of the proposed algorithm outperforms its counterparts by at least 13% in terms of flow time.

Index Terms—Satellite edge computing, big data analytics, query evaluation, online learning, approximation algorithms.

1 INTRODUCTION

1.1 Background

Streaming analytics that aims to process stream data is one major type of big data analytics. For example, media streaming is one of them, where objects with a continuous video stream need to be timely analyzed. But this is only

- The work by Zichuan Xu and Qiufen Xia was funded by the National Natural Science Foundation of China (NSFC) with grant numbers 62172068, 62172071, Natural Science Foundation of Liaoning (2023-MS-111), and the “Xinghai scholar” program. The work by Weifa Liang was supported by a grant from City University of Hong Kong with project No: 9380137/CS and No:7005845.
- Z. Xu, G. Xu, H. Wang (Corresponding author), and Qiufen Xia are with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian, China, 116621. E-mails: z.xu@dlut.edu.cn, 32017112@mail.dlut.edu.cn, haowang@dlut.edu.cn, qifenxia@dlut.edu.cn.
- W. Liang is with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: weifa.liang@cityu.edu.hk.
- S. Wang is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. Email: sqwang@bupt.edu.cn.

Manuscript received XXX; revised XXX.
1.2 Challenges
The challenges of timely evaluating big data queries of stream data in an SEC network are in the following.

The first challenge is that satellites move dynamically in their pre-setting orbits around the Earth. Unlike conventional mobile edge computing (MEC) networks on the ground that its edge servers are deployed at fixed locations, the edge servers (a.k.a satellites) in an SEC network move dynamically. Along with the dynamic service demand of ground users, this further complicates evaluation of big data queries of ground users. On one hand, the availability of satellites for receiving the streams of big data from ground users is very likely to be intermittent, due to not only unreliable wireless connections from ground users to satellites but also dynamic movement of the satellites. On the other hand, the connections among satellites (edge servers) are fully dynamic and may be intermittent as well. On receiving data streams from ground users, the collaboration among satellites is needed in order to jointly process the data streams. As such, the responsiveness of a query for big data analytics may be worsen if the connections from ground users to satellites and the collaboration among satellites are intermittent. Thus, how to minimize the flow time of queries, consisting of the time waiting for being scheduled, data transmission and processing delays, is challenging.

The second challenge is that the volumes of datasets generated by ground users are usually not known in advance. Along with the intermittent collaboration among satellites in an SEC network, the resources on satellites may become further fragmented, leading to queries being rejected or waiting for a prohibitive long time for evaluation. Specifically, the volume of a newly-generated dataset by an IoT device in a remote area is determined by unknown number of events happened and their appearance frequency in its environment. Along with the intermittent collaboration among satellites in the SEC network, careless query scheduling and assignment may lead to severely fragmented computing resources in satellites. The reason is that queries that cause resource fragments in a satellite cannot be migrated to other satellites if the connection among satellites are intermittent. Consequently, some queries may miss the opportunities of being evaluated in satellites with low processing delays. Thus, handling such an uncertainty in an SEC network with both intermittent connectivity and satellite collaboration does require careful query scheduling and assignment to avoid substantial wastage and fragmentation of limited computing resource in the network.

1.3 Motivation and Contribution
To address the afore-mentioned challenges, in this paper we investigate the flow-time minimization problem for big data analytics in an SEC network. Our motivations are in the following. First, the SEC network is fundamentally different from the conventional mobile edge computing (MEC) network. Existing techniques on big data analytics in conventional MEC networks cannot be directly applied to that in SEC networks. Specific reasons include (1) studies on conventional MEC networks usually consider the dynamics of mobile users and their intermittent connectivity to edge servers. Most of them assumed that edge servers or base stations in an MEC network are fixed, and their collaboration usually are stable [29], [31], [50], [53], [65]; (2) the fragmentation of edge resources is mainly due to uncertain user demands in conventional MEC networks. In contrast, in SEC networks, the uncertain user demand and intermittent edge collaboration jointly determines the severity of edge resource fragmentation. This further influences the timeliness of big data analytics in SEC networks. Second, although there are studies on architecture design [1], [8], [23], routing [46], [47], [68], and computation offloading [20], [21] in SEC networks, most of the studies are not applicable to timely big data query evaluation for streaming data analytics in SEC networks. Besides, there are rarely any studies on enabling timely query evaluations of big data analytics in SEC networks. The most closely studies of big data analytics in SEC networks are the ones in [15], [16]. Specifically, Huang et al. [15], [16] focused on collecting big data from ground users of the network such that the amount of data collected is maximized while the energy consumption of LEO satellites is minimized. They however did not consider timely processing of the collected data in the SEC network. To the best of our knowledge, we are the first to consider the flow-time minimization problem for big data analytics in an SEC network.

The main contributions of this paper are as follows.

- We formulate the flow time minimization problem for big data analytics in an SEC network, by formulating an Integer Linear Program (ILP) solution.
- We devise an approximation algorithm with an approximation ratio for the offline version of the flow time minimization problem, by proposing a novel technique of auxiliary graph construction.
- We propose an online learning algorithm with a bounded regret for the online flow-time minimization problem in an SEC network, with the assumption of uncertain data volumes of ground users.
- We evaluate the performance of the proposed algorithms, using real satellite networks. Simulation results show that the performance of the proposed algorithms outperform their counterparts.

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 introduces the system model, and defines the flow time minimization problem in an SEC network. Section 4 presents an exact solution to the offline version of the flow time minimization problem. Section 5 devises an approximation algorithm for the flow-time minimization problem in an SEC network, assuming that all requests are given. Section 6 proposes an online learning algorithm with a bounded regret for the online flow-time minimization problem in an SEC network. Section 7 evaluates the performance of the proposed algorithms, and the paper is concluded in Section 8.
the studies on big data analytics in both conventional MEC networks and SEC networks. We also summarize the differences of enabling big data query evaluation in the two types of networks.

2.1 Big Data Analytics in Conventional MEC Networks

Big data processing has gain much attention in ground MEC networks, and most of them focused on enabling timely big data processing, energy minimization of edge servers or age-aware big data query evaluations. However, most existing methods cannot be directly applied to the big data processing in SEC networks, because they usually do not consider dynamically available edge servers and uncertain volume of big data queries. For instance, Ndikumana et al. [34] investigated the problem of joint computing, caching, communication, and control in MEC networks with the objective to minimize a linear combination of bandwidth consumption and network latency, by adopting the block successive upper bound minimization method. Xu et al. [59] studied the problem of dynamic scheduling big data tasks in a cloud-edge computing environment, and their objective is to jointly optimize the execution time of tasks and the energy consumption of edge servers. Zhou et al. [67] focused on the problem of energy minimization of edge computing infrastructures by designing an architecture with node deployment, resource allocation and caching. Xia [54] investigated the problem of age-aware query evaluation for big data analytics in mobile edge clouds with the aim of enabling timely big data analytics, by adopting a novel metric named age of data and proposing an online learning based algorithm for the problem. They then studied the problem of proactive and intelligent query evaluation for big data analytics in edge clouds, by considering the materialized views of big data [55].

It must be mentioned that there are extensive work on task offloading in conventional MEC networks [45], [52], [57]. These studies cannot be directly applied to the big data processing in SEC networks, because they usually do not deal with the placement of big data and joint analysis on multiple datasets.

2.2 Big Data Analytics in SEC Networks

Low earth orbit (LEO) satellite networks are envisioned as a key technology to construct the satellite Internet for ubiquitous access of ground users on the Earth [4], [38]. Most existing studies aim to provide techniques and methods to integrate satellite and terrestrial networks to provide ubiquitous services for mobile users in remote regions of the Earth [56], [69], by focusing on the design of architectures [1], [8], [23], communication [43], and routing [46], [47], [68]. For example, Zhang et al. [69] proposed an architecture to integrate computing resources in both terrestrial and satellite networks via leveraging the technique of network function virtualization, and devised a task offloading method to achieve parallel processing in SEC networks. Chen et al. [8] studied the problem of controller placement and assignment in a software-defined LEO satellite network, by proposing an approximation algorithm with an approximation ratio. Pan et al. [47] compared the effect of establishing routing connections between different third-party relays and LEO satellite constellation on transmission delay in communication satellite networks. Tang et al. [46] used network coding to improve network throughput by mixing packets at intermediate nodes, and designed a multiple cooperative routing algorithms. Although the afore-mentioned studies enable the full integration of terrestrial and satellite networks, they usually ignore the provisioning of computational services in SEC networks.

To provide computational service in satellite networks, such as object detection and data stream processing, the SEC technique is investigated based on the integrated terrestrial and satellite networks. Specifically, computing resources of satellites in SEC environments are managed to implement various services and execute tasks from ground users. Studies related to SEC usually focus on service coverage and provisioning [9], [21], [27], task scheduling and offloading [21], or energy optimization of satellites [28], [41]. For example, Jia et al. [21] considered the service chaining problem in a software-defined satellite network, with the aim to minimize the resource consumption of terrestrial tasks by proposing both exact and approximation algorithms for the problem. Li et al. [28] focused on enabling sustainable satellite edge computing with the objective to extend battery lives of LEO satellites, by devising an online scheduling algorithm with a bounded regret. Also, Li et al. [27] investigated the service coverage problem in an SEC network, by proposing a service placement algorithm with a performance guarantee, through leveraging the Lyapunov optimization and Gibbs sampling techniques. Tang et al. [44] considered a three-tier computation architecture of an SEC network with both edge and cloud computing resources, and proposed a computation offloading algorithm for tasks of ground users such that the energy consumption of ground users is minimized. Although the afore-mentioned studies on SEC considered the scheduling and assignment of requests from ground users, they usually consider generic user requests/tasks with the given computing resource demands. Note that such generic user requests/tasks are fundamentally different with big data queries in the following aspects: (1) generic requests or tasks usually assume a given amount of computing resource demand, while ignoring the amount of data that needs to be processed by each request/task. Besides, each query needs jointly processing multiple datasets, which is ignored by generic requests/tasks; and (2) big data queries are sensitive to the flow time of an evaluation process, which is determined not only by the scheduling of requests but also by the locations at which the datasets of each query are located. Therefore, considering that the afore-mentioned fundamental differences, the methods for generic requests/tasks in MEC networks cannot be directly applied to problems arisen in SEC networks in this paper.

2.3 Differences and Motivations on the Design of New Big Data Query Evaluation in SEC Networks

There are fundamental differences between conventional MEC networks and SEC networks, existing methods cannot be directly applied to big data query evaluation in SEC networks. First, SEC networks are different with many ground networks, such as vehicular edge networks in terms of resilient requirements. In particular, applications in an
SEC network need to operate in a fully dynamic environment while continuing to provide continuous services with acceptable service quality levels. Specifically, the resources in satellites intend to be more fragmented than those in cloudlets on the ground, due to the direct connection between two satellites may be intermittent. The resilience to such intermittent connectivity is much more important than that of vehicular edge networks, considering that both resources and queries are dynamic. Even though there are studies on task offloading in vehicular edge networks with intermittent connection among users to road side units [2], [31], [40], [53], their methods cannot be directly applied to the problem in an SEC network, since the road-side units are located at fixed locations. Second, the big data queries in SEC networks have complex characteristics of data and user demand distributions. In particular, a big data query may need to consider both spatially and temporally distributed data. To obtain a good performance, the runtime of a big data processing system must be re-optimized for SEC networks. Besides, big data queries in SEC networks have uncertain user demand distributions. Considering the fact that big data queries need evaluating multiple datasets at different locations and the data volumes of such datasets are uncertain, the proposed methods thus cannot be directly applied. Studies on SEC networks rarely consider big data applications, and they only focused on generic requests/tasks.

3 PRELIMINARIES

We first describe the system model, notions and notations. We then define the optimization problems precisely.

3.1 System Model

We consider a satellite edge computing (SEC) network \( G = (\mathcal{S} \cup \mathcal{D}, E) \), consisting of a set \( \mathcal{S} \) of LEO satellites in the space and a set \( \mathcal{D} \) of data centers in the ground. The LEO satellites in \( \mathcal{S} \) are distributed in an orbital shell of the space around the Earth. Let \( s_k \) be a satellite in \( \mathcal{S} \), which is equipped with radio transmitters and receivers to communicate with ground stations. Each LEO satellite \( s_k \) is usually small in size and thus has limited computing capacity, including servers, GPUs, or FPGAs, to process big data of ground users [5], [10], [27], [44], [56], [69]. Following the settings of popular LEO satellite constellations, including Starlink, Kuiper and Iridium [6], [14], [42], satellites in \( \mathcal{S} \) are distributed into multiple inclined orbits, and the satellites in each orbit are evenly distributed, circulating along their orbit periodically [18]. In a constellation of LEO satellites, intersatellite links (ISLs) based on laser transmitters and receivers are used to support communications among the satellites in each constellation [9], [23], [44], [46]. Following the well known topology +Grid for constellations, we assume that each satellite connects to two adjacent satellites in the same orbit and two adjacent satellites in adjacent orbits [17], [27], [36], [39]. Besides, satellites that are not within the transmission range of each other in a constellation can communicate via multiple hop ISLs [44].

Each LEO satellite circulates periodically around the Earth in its orbit at a constant speed. The current location of the satellite determines the set of ground users it can cover. Without loss of generality, we assume that the length of each time slot \( t \) is small enough that the set of covered ground users does not change in each time slot. Assuming that a finite time horizon \( T \) is equally divided into multiple time slots. We further assume that the length of each time slot is far less than the longest communication time of each ground user \( u_i \) by submitting his/her big data analytical query to the SEC network. Thus, the evaluation of a query may span multiple time slots of the finite time horizon \( T \).

Ground users, such as different IoT devices, home users, or vehicles, access the SEC network via different types of ground stations. Denote by \( u_i \) a ground user requiring big data processing in the SEC network. We assume that each data center and each ground user \( u_i \) can communicate with LEO satellites via a ground station. Fig. 1 gives an example of the SEC network.

![Fig. 1. An example of an SEC network \( G = (\mathcal{S} \cup \mathcal{D}, E) \), provisioning query services for big data analytics.](image)

3.2 Big Data Query Evaluation in an SEC Network

To unveil valuable insights behind the big data generated by ground users, queries are usually issued to analyze the data in the SEC network. To this end, a ground user in a remote area submits a big data analytical query to an LEO satellite in the SEC network when the LEO satellite circulates and the ground user is under its coverage. Specifically, ground user \( u_i \) may continuously generate datasets and issue big data analytical queries to analyze its newly-generated and previously placed datasets within a finite time horizon \( T \). Denote by \( r_m \) a big data analytical query of ground user \( u_i \) and its generation time is represented by \( \tau_m \) within time horizon \( T \). Let \( ds_m \) be the dataset that is generated by ground user \( u_i \) in time slot \( \tau_m \). A big data analytical query is issued as soon as the dataset generated in each time slot, and each query \( r_m \) may require to analyze the datasets that are already placed into the SEC network. Let \( D_{S_m} \) be the set of datasets for the evaluation of query \( r_m \). Each big data analytical query \( r_m \) can be represented by a quadruple \( r_m = (u_i, \tau_m, ds_m, D_{S_m}) \).

Each big data analytical query \( r_m \) can be evaluated either in LEO satellites or in a data center if the LEO satellites do not have sufficient available computing resource, we distinguish this into the following two cases.
3.3 Flow-Time of Big Data Query Evaluation

Once query \( r_m \) is scheduled for evaluation in the current time slot \( t \), it can send its data stream to an LEO satellite within its transmission range until its amount \( y_{m,t,k} \) of a proportion of dataset \( ds_m \) is uploaded. Meantime, the LEO satellite evaluates the query until all of its assigned data volume is processed. We assume that the processing result of each big data analytical query is small in size, and the time of returning results to the ground user can be ignored. Let \( C_m \) be the time slot when the system completes the processing of big data analytical query \( r_m \). The flow time of query \( r_m \) is

\[
C_m - \tau_m + 1,
\]

assuming that query \( r_m \) can only be processed in the next time slot of its arrival time slot. According to the evaluation process of each query \( r_m \), we divide the flow time of query \( r_m \) into the response delay, data uploading delay, and evaluation delay, which are defined as follows.

**Response delay:** The response delay of query \( r_m \) is defined as the duration between the time when it is responded and its arrival time. Specifically, when query \( r_m \) arrives at the system, we need to decide when it will be evaluated in the current time slot \( t \). In particular, the evaluation of query \( r_m \) may be postponed when there is no satellites within its transmission range. Also, when its current visible satellites are overloaded, the query is needed to be scheduled later for evaluation to save prohibitive processing delays in the overloaded satellites. We thus use \( x_m \) to indicate in which time slot query \( r_m \) is responded.

If \( x_m = t \), query \( r_m \) is responded in time slot \( t \), and the datasets required by \( r_m \) will be processed in the following time slots starting at time slot \( t \). Therefore, the response delay of query \( r_m \) is

\[
x_m - \tau_m + 1.
\]

**Data uploading delay:** Assume that query \( r_m \) sends its generated dataset \( ds_m \) to satellite \( s_k \) for processing at time slot \( t \), the delay incurred on the data uploading process refers to as the data uploading delay. Note that the data of \( r_m \) is uploaded first to its closest satellite that is within its transmission range, and then is forwarded to \( s_k \) via ISL links in the SEC network. Let \( R_{i,t} \) be the achieved data rate \( R_i \) (bits per second) via the wireless channel from ground user \( u_i \) to its closest satellite for request \( r_m \) in time slot \( t \), which can be calculated by well-known wireless communication model with a given amount of bandwidth, channel gain of each ground user and its transmission, noise and interference powers [57].

Note that the communication between an LEO satellite and a ground station are based on Ka or Ku frequency band, the achieved data rate \( R_{i,t} \) may be affected by the communication distance and the rain attenuation. However, considering that the evaluation of a big data query usually does not last long, we assume that the meteorological environment remains stationary during the big data query evaluation. As such, the data rate will not be affected by the rain attenuation when a query is evaluated [49], [66].

Denote by \( \eta_{m,k,t} \) the delay of uploading a unit of dataset of each query \( r_m \) to satellite \( s_k \) in time slot \( t \), then

\[
\eta_{m,k,t} = \frac{D_{\text{unit}}}{R_{i,t}} + \sum_{p_{m,k}} d_c \cdot D_{\text{unit}},
\]

where \( p_{m,k} \) is the path from query \( r_m \)’s closest satellite that is in its transmission range to the selected satellite \( s_k \).

Besides, query \( r_m \) may demand the other datasets in
where $p_{k',k}$ is a shortest path from $s_{k'}$ to $s_k$ and $d_e$ is the delay of transmitting a unit amount of data via an ISL $e \in p_{k,k}$.

If query $r_m$ is to be evaluated in a data center $DC_j$, its dataset $d_{s_m}$ needs to be forwarded from satellite $s_k$ to data center $DC_j \in DC$. The data uploading delay then consists of the delay of $\eta_{m,k}$ and the delay of forwarding both the starting units of both the newly-generated dataset $d_{s_m}$ and the datasets in $DS_{s_m}$ from satellites to data center $DC_j$. The delay of transmitting a unit amount of dataset $d_{s_m}$ from a satellite $s_k$ to a data center $DC_j$ can be calculated by Eq. (3), which is denoted by $\eta_{k',j}$.

Note that query $r_m$ that is scheduled for evaluation may be interrupted for later evaluation. The data uploading process to its previous satellite will be interrupted as well. Therefore, the data uploading needs to be forwarded to a new satellite within the transmission range of its user $u_i$. If query $r_m$ is re-scheduled for evaluation from satellite $s_k$ to another satellite $s_{k'}$, its data needs to be forwarded from its user $u_i$ to satellite $s_{k'}$. This incurs an additional data uploading delay of $\eta_{m,k'}$.

**Evaluation delay:** Once the starting units of the datasets of query $r_m$ is transferred to its assigned satellite $s_k$, the query will be evaluated immediately, by analyzing the streamed datasets continuously. Note that the complexity and delay of evaluating each big data query largely rely on both the quality and volume of datasets. Specifically, more computing resource may be needed to process a dataset with low quality compared with a dataset with the same volume and high quality, following many existing studies [3], [19], [63]. We thus calculate the delay $\eta_{m,k'}^{proc}$ of evaluating query $r_m$ in satellite $s_k$ by

$$
\eta_{m,k'}^{proc} = \text{data size} \times \text{processing rate} = \sum_{d_{s_m} \in DS_m \cup \{d_{s_m}\}} |d_{s_m}| \cdot \beta_k,
$$

where $\beta_k$ is a constant identifying how long satellite $s_k$ takes to process a unit amount of data. Similarly, the evaluation delay of $r_m$ in a data center $DC_j \in DC$ is

$$
\eta_{m,k}^{proc} = \sum_{d_{s_m} \in DS_m \cup \{d_{s_m}\}} |d_{s_m}| \cdot \beta_j,
$$

where $\beta_j$ is the time data center $DC_j$ for processing a unit amount of data.

Note that the flow-time is part of the quality of service of many services. An acceptable flow-time of different queries varies according to the type of their applications. For example, aerospace big data queries require real-time implementation, because real-time transmission of spacecraft telemetry data and flight data is required for flight control and monitoring. On the other hand, Earth monitoring big data queries need to be responded as soon as possible to make sure the results are timely, not necessarily in real-time.

### 3.4 Resource Consumption of Satellites and Data Centers

Following existing studies [3], [20], [63], we assume that the datasets of queries can be processed within a given time duration of $d$ by assigning the amount $\xi$ of computing resource for the processing of a unit amount of data. To speed up the processing, the satellite can assign more computing resources to process the data. Specifically, let $\Delta$ be the length of each time slot $t$, if satellite $s_k$ wants to process amounts $y_{m,k} \in DS_m$ of data for query $r_m$ within $\Delta - (\sum_{i=1}^{k} \eta_{m,i}^{proc} + \eta_{m,k}^{upload})$, the amount of computing resource needed is

$$
C(y_{m,k}) = \xi \cdot y_{m,k} (d_i/(\Delta - (\sum_{i=1}^{k} \eta_{m,i}^{proc} + \eta_{m,k}^{upload}))).
$$

Let $CR_k$ be the capacity of computing resource on satellite $s_k$. We have

$$
C(y_{m,k}) \leq CR_k.
$$

Since data centers usually have abundant computing resource that can be virtualized into VMs, we assume that each query is assigned to a VM with abundant computing resource if it is assigned to a data center. Let $CR_j$ be the computing resource available to a VM in data center $DC_j$.

### 3.5 Problem Definitions

Given an SEC network $G = (S \cup DC, E)$ with a set $S$ of LEO satellites and a set $DC$ of data centers in the ground, a set of ground users who have queries for big data analytics be evaluated in either the LEO satellites or the data centers, assume that a ground user issues a query $r_m$ in time slot $t$ to analyze a collection of datasets that consist of a newly-generated dataset $d_{s_m}$ and multiple historical datasets in $DS_m$ that are placed in the SEC network already. We consider the following optimization problems.

Given a set $R$ of the arrived queries, the **flow time minimization problem for big data analytics in an SEC network** is to evaluate each query $r_m \in R$ in either a satellite or a data center of $G$, by uploading the newly-generated dataset $d_{s_m}$ for $r_m$ to a selected satellite $s_k$, transmitting the required historical datasets in $DS_m$ of $r_m$ from other satellites to the selected satellite $s_k$ (or the selected data center), such that the average flow time of all query evaluations is minimized, subject to the resource capacity on each satellite $s_k \in S$. Note that query execution is allowed to be interrupted and resumed later in other satellites if the selected satellite for the query evaluation is out of the transmission range of the query user or overloaded.

In reality, user queries usually arrive into the system one by one without the knowledge of their future arrivals and the volume of the newly-generated dataset of each query is not known in advance, the **online flow time minimization problem for big data analytics in an SEC network** is to evaluate each query $r_m$ in either a satellite or a data center dynamically, by uploading the newly-generated dataset $d_{s_m}$ of $r_m$ to a selected satellite $s_k$, transmitting its historical datasets in $DS_m$ from other satellites to satellite $s_k$ (or the selected data center), such that the average flow time of all queries is minimized, subject to resource capacity on each satellite $s_k \in S$.

For clarity, the symbols used in this paper are summarized in Table 1.
4 AN EXACT SOLUTION TO THE FLOW TIME MINIMIZATION PROBLEM

In this section we formulate an ILP solution to the flow time minimization problem for big data analytics in an SEC network as follows. Let $S_{m,t}$ be the set of satellites within the transmission range of ground user of query $r_m$, which can be determined as a priori assumption that satellites are circulating the Earth periodically. The objective of the problem thus is

\[
\text{ILP: min } (\sum_{r_m \in R} (C_m - \tau_m + 1))/|R|.
\]

Recall that $y_{m,t,k}$ indicates the amount of a proportion of dataset $d_{s_m} \in D_{S_m} \cup \{d_{s_m}\}$ is processed in satellite $s_k$ at time slot $t$, and $y_{m,t,j}$ is a decision variable indicating the amount of a proportion of dataset $d_{s_m} \in D_{S_m} \cup \{d_{s_m}\}$ that is processed in data center $DC_j$ at time slot $t$. For clarity, we use $x_{m,t}$ and $x'_{m,t}$ to denote binary indicator variables to determine whether query $r_m$ will be scheduled in a satellite or a data center to process a portion of its data at time slot $t$, respectively. We have $C_m = \max\{\{t \cdot (x_{m,t} + x'_{m,t})\}$.

Assuming that $d_{\text{max}} = \max\{d_{s_m}\}$, the constraints of the ILP formulation is given as follows.

\[
\sum_{t \geq \tau_m} \sum_{k,j} y_{m,t,k} + y_{m,t,j} = |d_{s_m}|, \forall r_m
\]

\[
\sum_{t \geq \tau_m} \sum_{k,j} y_{m,t,k} + y_{m,t,j} = |d_{s_m'}|, \forall r_m, d_{s_m'} \in D_{S_m}
\]

\[
\sum_{r_m \in R} \sum_{t \geq \tau_m} \sum_{k,s,k,s \in S_{m,t}} \frac{\xi \cdot y_{m,t,j} \cdot d}{(\Delta - \eta_{m,k,t} - \eta_{k,j})} \leq CR_k, \forall t, DC_j \in DC
\]

where Constraint (10) says that the newly-generated dataset $d_m$ of query $r_m$ at time slot $t_m$ needs to be analyzed in the following time slots starting from $\tau_m$. Constraint (11) ensures that the historical datasets in $D_{S_m}$ required by query $r_m$ need to be analyzed in either a satellite or a data center. Constraint (12) indicates that the computing resource capacity on each satellite $s_k$ cannot be violated. Note that the LHS of Ineq. (12) includes the time spent for uploading a unit amount of data $D_{\text{unit}}$ of $r_m$ to enable subsequent data processing at time slot $t$. Intuitively, if the transmission of a unit amount of data $D_{\text{unit}}$ takes longer within time slot $t$, then less time will be spent for the processing of amounts $y_{m,t,k}$ of data. This means that more computing resource is needed to speed up the process to ensure that the amount $y_{m,t,k}$ of data is processed within the time length $\Delta$ of each time slot $t$. Constraint (13) implies that the resource capacity on a VM in each data center $DC_j$ should not be violated. Constraints (14) and (15) ensure that both $x_{m,t}$ and $x'_{m,t}$ are 1s if there are assigned data of $r_m$ for processing at time slot $t$. Constraint (16) ensures that $x_{m,t}$ and $x'_{m,t}$ are 0s if they are not assigned data of $r_m$ for processing at time slot $t$.
t in a satellite or a data center, respectively. Constraint (16) says that a query can only be assigned to either a satellite or a data center in each time slot. Constraint (17) guarantees \( x_{m,t} \) is a binary variable, Constraints (18) and (19) ensure that the amounts of datasets to be processed in each time slot are integers.

5 APPROXIMATION ALGORITHM FOR THE FLOW-TIME MINIMIZATION PROBLEM

We now propose an approximation algorithm with a provable approximation ratio for the flow-time minimization problem in an SEC network.

5.1 Overview of the Algorithm

The basic idea of the approximation algorithm is to reduce the flow time minimization problem in an SEC network to the minimum-cost multicommodity flow problem in an auxiliary graph that is derived from the SEC network \( G \). Recall that in the flow-time minimization problem, we schedule the execution of each query \( r_m \) into different time slots after its arrival. Also, the satellites that can cover the ground user of query \( r_m \) dynamically change. However, in a conventional flow network, the features of dynamic availability of satellites and dynamic arrivals of queries are not considered. We adopt a fully-customized auxiliary construction technique, by merging both spatial and temporal availabilities of satellites in the SEC network. Specifically, we merge the temporal availability of satellites and dynamic arrivals of requests, by adopting a concept of time widget with each consisting of a set of satellites that are available for queries at each time slot. Based on the constructed auxiliary graph, we then find a splittable flow in the auxiliary graph. Although a dataset can be split and allocated to different time slots, it cannot be allocated to multiple satellites in the same time slot. We then merge the split flows at different satellites at each time slot into a single satellite or a single data center. By merging the flows carefully, we make sure that the obtained solution is a feasible solution that is near to the optimal one.

5.2 Approximation Algorithm for the Problem with Uniform Data Size

The proposed approximation algorithm consists of two stages: stage 1, constructing an auxiliary graph \( G' = (V', E') \); and stage 2, finding a minimum cost flow in \( G' \) and deriving a feasible solution to the online flow-time minimization problem from the found flow.

We first describe stage 1 of the proposed approximation algorithm. Specifically, the auxiliary graph \( G' \) is constructed as follows. Each query \( r_m \) can only be scheduled for evaluation in time slots no earlier than its arrival time slot \( \tau_m \). The resource availability of satellites and data centers in time slots no earlier than \( \tau_m \) varies. For simplicity, we consider a satellite or a data center as a potential location for query \( r_m \), and denote by \( L_o \) the potential location, where \( L_o \in \{S \cup DC\} \) with \( 1 \leq o \leq |S \cup DC\} \).

To incorporate such dynamics, we create \( 2 \cdot T \) copies for each location in \( \{S \cup DC\} \), which are referred to as virtual locations. We then add all the virtual locations of each \( L_o \in \{S \cup DC\} \) into \( V' \). Specifically, each location \( L_o \) has two virtual locations in time slot \( t \), which are denoted by \( L_{o,t} \) and \( L_{o,t}' \), respectively. Then, \( V' \leftarrow V' \cup \{L_{o,t}, L_{o,t}'\} \). We consider all the virtual locations of a time slot as a time widget, which represents the status of all potential locations to evaluate queries in that time slot. An example of the time widget is shown in Fig. 2. In addition, a query node for each query \( r_m \in R \) is added too, i.e., \( V' \leftarrow V' \cup \{r_m\} \). There is also a virtual sink node in the auxiliary graph, denoted by sink.

We now consider the construction of edges in \( G' \). We first connect each query node \( r_m \) with each virtual location \( L_{o,t} \) if \( \tau_m \leq t \), i.e., \( E' \leftarrow E' \cup \{(r_m, L_{o,t})\} \). The rationale behind is that the query \( r_m \) can only be evaluated after its generation at time slot \( \tau_m \). The capacity of each of such edges is set to

\[
u(r_m, L_{o,t}) = |DS_m| + 1,
\]

implying the number of datasets to be processed by query \( r_m \). The cost of each of such edges is set to

\[
c(r_m, L_{o,t}) = t - \tau_m + 1.
\]

We then connect the virtual locations in each time widget. Specifically, there is a directed edge \( (L_{o_1,t}, L_{o_2,t}) \) from virtual location \( L_{o_1,t} \) to virtual location \( L_{o_2,t}' \) and the capacity of the edge corresponds to the number of datasets that will be evaluated at location \( L_o \) (a satellite or a data center). Recall that we here assume that all datasets have the same volume, which is denoted by \(|ds|\). Also, the data transmission of each dataset takes time. Therefore, we set

\[
u(L_{o_1,t}, L_{o_2,t}') = \left[ (CR_k \Delta - \eta_{min}) / (\xi \cdot d \cdot |ds|) \right],
\]

where \( \eta_{min} = \min\{\min_{m,k,t} \eta_{m,k,t}, \min_{k,j} \eta_{k,j}\} \). The cost of edge \( (L_{o_1,t}, L_{o_2,t}') \) is set to zero. We also connect the virtual locations in each time widget with node sink. That is, there is an edge from every \( L_{o,t} \) to sink, i.e., \( E' \leftarrow E' \cup \{L_{o,t}, \text{sink}\} \). The capacity on the edge is set to infinity, and its cost is set to zero. Fig. 2 shows an example of the constructed auxiliary graph \( G' \).

We then describe stage 2 of the proposed approximation algorithm. We now find a minimum cost flow in \( G' = (V', E') \). Specifically, we treat each query \( r_m \) as
Algorithm 1 An approximation algorithm for the flow-time minimization problem for big data analytics in an SEC network, i.e., ApproUN.

**Input:** An SEC network $G = (S \cup DC, E)$, a set $S$ of LEO satellites and a set $DC$ of data centers in the ground, a set of ground users, each query $r_m$ of a ground user usually is issued in a time slot $\tau_m$ to analyze its required set of datasets with a uniform data size.

**Output:** A scheduling and resource allocation strategy for each query.

1. for each query $r_m$ do
2. Divide each dataset $ds_m \in DS_m \cup \{ds_m\}$ of query $r_m$ into $\lceil |ds_m| / |ds| \rceil$ data splits;
3. Invoke Algorithm ApproUN to obtain a solution with each query having a set of datasets with a uniform data size of $|ds|$
4. For each time slot $t$, if a satellite $s_k$ is assigned with $n_{k,t}$ data splits. We move the data splits in each time slot $t$ to the satellite with the most data splits, for each query $r_m$;
5. return The scheduling and resource allocation for each $r_m$;

Algorithm 2 An approximation algorithm for the flow-time minimization problem for big data analytics in a SEC network, i.e., ApproUN.

**Input:** An SEC network $G = (S \cup DC, E)$, a set $S$ of LEO satellites and a set $DC$ of data centers in the ground, a set of ground users, each query $r_m$ of a ground user usually is issued in a time slot $\tau_m$ to analyze its required set of datasets with different data sizes.

**Output:** A scheduling and resource allocation strategy for each query.

1. for each query $r_m$ do
2. Divide each dataset $ds_m \in DS_m \cup \{ds_m\}$ of query $r_m$ into $\lceil |ds_m| / |ds| \rceil$ data splits;
3. Invoke Algorithm ApproUN to obtain a solution with each query having a set of datasets with a uniform data size of $|ds|$
4. For each time slot $t$, if a satellite $s_k$ is assigned with $n_{k,t}$ data splits. We move the data splits in each time slot $t$ to the satellite with the most data splits, for each query $r_m$;
5. return The scheduling and resource allocation for each $r_m$;

a maximum ratio of $1-\alpha/1-\beta$ in the worst case, where $\alpha = \max_{m,k,t} \eta_{m,k,t}, \eta_{m,k,t} = \{\max_{m,k,t} \eta_{m,k,t}, \max_{m,k,t} \eta_{m,k,t}, \min_{m,k,t} \eta_{m,k,t}, \min_{m,k,t} \eta_{m,k,t}\}$, $\eta_{m,k,t} = \{\min_{m,k,t} \eta_{m,k,t}, \min_{m,k,t} \eta_{m,k,t}\}$.

Please see Appendix for the detailed proof.

**Theorem 1.** Given an SEC network $G = (S \cup DC, E)$ with a set $S$ of LEO satellites and a set $DC$ of data centers in the ground, a set $R$ of ground users, each query $r_m$ of a ground user usually is issued in a time slot $\tau_m$ to analyze its required set of datasets consisting of both a newly-generated dataset $ds_m$ and previously-generated datasets in $DS_m$ that are already placed in the SEC network. Assuming that all datasets have a uniform size, there is an approximation algorithm, Algorithm ApproUN, with an approximation ratio of $O(T^2)$, which delivers a feasible solution to the flow-time minimization problem for big data analytics in $G$, where $T$ is the length of a finite time horizon.

Please see Appendix for the detailed proof.

**Theorem 2.** Given an SEC network $G = (S \cup DC, E)$ with a set $S$ of LEO satellites and a set $DC$ of data centers in the ground, a set $R$ of ground users, each query $r_m \in R$ of a ground user usually is issued at time slot $\tau_m$ to analyze its required set of datasets consisting of both a newly-generated dataset $ds_m$ and previously-generated datasets in $DS_m$ that are already placed in the SEC network. Algorithm ApproUN delivers a feasible solution to the flow-time minimization problem for big data analytics in $G$ with approximation ratio $O(T^2)$ for big data analytics, at the expense of resource capacity violation on each satellite by a maximum ratio of $\min\{1-\alpha, 1-\beta\} \lceil |ds_{\text{max}}| / |ds| \rceil$, where $T$ is the length of a finite time horizon.

Please see Appendix for the detailed proof.

5.3 Approximation Algorithm to the Problem with Diverse Data Volumes of Datasets

So far we assumed that the sizes of all datasets for a query are the same. We now remove this assumption and propose another approximation algorithm with an approximation ratio for the flow time minimization problem in an SEC network.

The basic idea of the algorithm is to partition each dataset into a number of data splits with the same size except the last one. The algorithm then adopts the proposed algorithm ApproUN. However, the obtained solution may be infeasible to the flow time minimization problem, since the data splits of each dataset may be assigned to different satellites at each time slot. To achieve a feasible solution, we merge each data split of a dataset allocated to different satellites to a single satellite. Specifically, let $|ds|$ be the size of each data split.

Each dataset $ds_m$ can be divided into $\lceil |ds_m| / |ds| \rceil$ data splits. The given data splits of the datasets of each query, we then invoke Algorithm ApproUN to obtain a solution. Assume that at each time slot $t$, a satellite $s_k$ is assigned with $n_{k,t}$ data splits. For each query $r_m$, we move the data splits at each time slot $t$ to the satellite with the most data splits.

The detailed algorithm is shown in Algorithm 2, which is referred to as Algorithm ApproUN.

5.4 Algorithm Analysis

In the following, we analyze the feasibility and performance of the proposed algorithms ApproUN and ApproUN.

**Lemma 1.** The solution delivered by Algorithm ApproUN violates the computing resource capacity of each satellite by a maximum ratio of $\alpha$.

6 Online Learning Algorithm for the Online Flow-Time Minimization Problem

We now consider the online flow-time minimization problem for big data analytics in an SEC network, where queries for big data analytics arrive one by one. For each arrived query $r_m$, the data volume of its newly-generated dataset $ds_m$ may...
not be given in advance. We propose an online learning algorithm with a bounded regret for the problem.

### 6.1 Design Rationale

Our basic idea behind the proposed algorithm is to invoke Algorithm Appro in the beginning of each time slot $t$ that splits each dataset into a number of data splits with each having size of $|ds|$. We observe that the choice of the size of each data split is vital for the performance of Appro. Specifically, given a large value of $|ds|$, each dataset is split into a small number of data splits. Thus, the size of the last data split of each dataset may be much smaller than $|ds|$. This unfortunately may waste the computing resource of satellites, due to the internal resource fragmentation in the last data split of each dataset. Eventually, such resource fragmentation forces more queries to be rejected due to the lack of resource in satellites. On the other hand, if a smaller $|ds|$, the resource wastage may be reduced, the efficiency of the algorithm is reduced. The rationale behind is that more commodities are created in Algorithm Appro and dealing with more number of commodities require more time. Also, the resource violation ratio is higher as indicated in Theorem 2. To address the above-mentioned issue, we adopt an online learning method to dynamically learn a proper value of the size of each data split in each time slot. Also, the resource violation ratio is higher as indicated in Theorem 2. To address the above-mentioned issue, we adopt an online learning method to dynamically learn a proper value of the size of each data split in each time slot. Also, the resource violation ratio is higher as indicated in Theorem 2. To address the above-mentioned issue, we adopt an online learning method to dynamically learn a proper value of the size of each data split in each time slot.

A scheduling and resource allocation strategy for each query.
1. Divide the interval $Z$ into $\kappa$ intervals with fixed length $\varepsilon = \frac{|ds_{\text{max}}| - |ds_{\text{min}}|}{\kappa}$, and let $Z'$ be the discretized set of arms;
2. for each time slot $t \leftarrow 1 \cdots T$
   3. Let $R'_t$ be the set of arrived requests in the beginning of $t$;
   4. Try all active arms in $Z'$ in possibly multiple rounds;
   5. for each arm $a \in Z'$ do
      6. if there exists an arm $a'$ with $UCB_t(a) < LCB_t(a')$
         then
            7. Deactivate arm $a$;
            8. Choose an active arm in $Z'$ that has the maximum reward, and use its value as the value of $|ds|$;
      9. Invoke Algorithm Appro with $R'_t$ and the selected value for $|ds|$;
   
   by $R^*(Z')$ the best arm in $Z'$ that achieves the maximum reward of responding queries.

To maximize the reward of the learning approach, we eliminate the arms that may produce low rewards progressively. To this end, we set all arms in $Z'$ to the 'active' states. To this end, let $UCB_t(a) = R_t(a) + r_t(a)$ and $LCB_t(a) = R_t(a) - r_t(a)$ be the upper and lower bounds on the reward of arm $a$ at time slot $t$, where $r_t(a)$ is the confidence radiation. We deactivate all arms $a$ if there is an arm $a' \in Z'$ with $UCB_t(a) < LCB_t(a')$. This procedure continues until only one arm is left in $Z'$. We finally select the last arm as the value of $|ds|$ in time slot $t$. Given the selected value of $|ds|$, the detailed online learning algorithm for the online time-flow minimization problem is given in Algorithm 3, which is referred to as $OL$.

### 6.2 Online Learning Algorithm

Finding an appropriate value of $|ds|$ is challenging, since there are infinite numbers of such values in the range of $Z = [|ds_{\text{min}}|, |ds_{\text{max}}|]$. To tackle this challenge, we discretize the continuous range $Z$ into a finite set of candidate values for $|ds|$. Given the candidate set of values for $|ds|$, we then select a value following a customized Lipschitz bandit learning method. Specifically, given the range $Z$ of $|ds|$, we treat each value as an ‘arm’, and the reward of selecting an arm in $Z$ varies. We define the inverse of the average flow time of all queries that are responded according to the selected value of $|ds|$ as the reward. Let $R(a)$ be the expected reward of selecting an arm $a$ in $Z$, and we further assume that the reward satisfies the following Lipschitz condition.

$$ |R(a) - R(b)| \leq \psi |a - b|, \text{ for any arms } a, b \in Z, $$

where $\psi$ is a given constant.

We divide the range $Z$ into $\kappa$ subintervals with fixed length $\varepsilon = \frac{|ds_{\text{max}}| - |ds_{\text{min}}|}{\kappa}$, so that the obtained set $Z'$ consists of all integers multiplying by $\varepsilon$. Selecting different values for the arms in $Z'$ obtains different rewards. Denote

### 7 Experiments

In this section we evaluate the performance of the proposed algorithms through conducting experiments, by simulations in an SEC network with a real network topology.
7.1 Simulation Settings

We consider an SEC network with 60 LEO satellites that are distributed into 6 orbits, with 10 LEO satellites in each orbit, following the structure and settings of the well-known Iridium constellation [17]. The height of each orbit is 780 kilometers, and the orbital inclination angle of each orbit is set to 86.4 degrees [17]. Given the limited size of each LEO satellite, we consider that each satellite is attached with lightweight accelerators, such as NVIDIA Jeston Orin NX [33].

The computing resource capacity of each satellite thus is randomly drawn within the range of [1, 000, 3, 600] MHz [5, [33]]. There are 20 data centers that are distributed according to the locations from real datasets in [23]. The amount of computing resource leased by each data center to implement queries is generated from [2, 000, 3, 600] MHz [5], which is equivalent to the computing capacity of several virtual machines. The transmission power of each ground user is withdrawn from [18, 23] dBm [11] and the channel bandwidth is generated within the range of [16, 20] Mbps [44]. The length of each time slot is 0.05 seconds [58]. The number $\kappa$ of subintervals of Algorithm OL is set to 8.

We consider big data queries in industrial IoT environments. Such queries are issued to enable real-time analysis on environmental temperatures, pressures, and other sensor readings that captures the statuses of large industrial equipments including wind turbines, compressors, motors, and pumps. We thus consider that the size of each dataset is randomly drawn from $[1, 000, 6, 000]$ Kilo Bytes (KB) [7], [32], [44], [62]. The number of datasets required by each query is randomly drawn from 1 to 5. The volume of $D_{unit}$ is set to 200 KB.

We compare the proposed algorithms against the following benchmarks:

- An algorithm that schedules each query immediately after its arrival (with zero response delay) by greedily selecting a satellite with the minimum data uploading delay from a set of satellites within the transmission range of a ground user and have enough available computing resource to evaluate the query at this moment. Note that this benchmark is widely adopted by previous studies in satellite-terrestrial communication networks [8], [26], [60]. For simplicity, this benchmark is referred to as algorithm Greedy;

The online version of this benchmark is referred to as Greedy_OL. Specifically, algorithm Greedy_OL schedules each arrived query immediately according to the resource availability of at the current time slot.

- A special version of algorithm Appro without considering the response delay of each query, which is referred to as Appro_noRes. Specifically, algorithm Appro_noRes does not consider the arrival times of queries when scheduling them. It only minimizes the sum of data uploading delays and evaluation delays.

- A delay-oriented task scheduling algorithm in a space-air-ground network [66], with the aim of minimizing the processing latency of tasks (corresponding to the evaluation delay of queries). The algorithm is based on a Q-learning process that adopts a deep neural network (DNN) to find the optimal Q-value. For simplicity, this algorithm is referred to as DOTS.

- A multi-hop computation offloading algorithm in a vehicular fog computing (VFC) network [31], referred to as algorithm CODE-V. Note that VFC network is a terrestrial edge computing network with intermittent connections. Algorithm CODE-V is based on a differential evolutionary process, and aims to choose the optimal client vehicle task-to-fog assignment decision to reduce end-to-end service delay. Note that CODE-V is not particularly designed for SEC networks. It thus does not consider the mobility of edge servers. We modify the algorithm for SEC networks by considering each satellite as an edge server. It must be mentioned that this benchmark is used to demonstrate the need of new algorithms for SEC networks.

The metrics adopted in evaluating the proposed algorithms against their benchmarks include the average flow time, the resource violation ratio, and the running time of algorithms. The optimization objectives of the flow time minimization problem and the online flow time minimization problem are to minimize the flow time while guaranteeing the resource constraints. Note that algorithm TLP obtains an exact solution that meets resource constraints, and thus the obtained solution does not violate resource capacities. We thus do not show the resource violation ratios of TLP. Similarly, algorithm Greedy does not violate the resource capacities on satellites as it selects from the satellites with suf-
7.2 Evaluation of Algorithms for the Flow Time Minimization Problem

We first evaluated the performance of algorithms ILP, Appro, Appro_UN, Greedy, and Appro_noRes for the flow time minimization problem in terms of the average flow time, computing resource violation ratio, and running time of algorithms, by varying the maximum size of each dataset from 2,000 KB to 6,000 KB while keeping the number of queries at 10. We can see from Fig. 4 (a) that algorithms Appro and Appro_UN have the lower average flow times than those of algorithms Greedy and Appro_noRes. Specifically, Appro delivers a 22.5% lower flow time than that of algorithm Greedy. The reason is that Appro and Appro_UN minimize the average flow time by striving for a nice trade-off between the response, transmission and processing delays. However, algorithm Greedy only considers the response delay while algorithm Appro_noRes deals with transmission and processing delays without response delays. This demonstrates the importance of flow time minimization.

In addition, algorithm Appro_UN has a lower average flow time than that of algorithm Appro, because Appro_UN deals with a special case of the flow time minimization problem where all datasets having the uniform size. Also, algorithm ILP has the minimum average flow time since it delivers an optimal solution to the problem. From Fig. 4 (b), we can see that the proposed algorithms Appro, Appro_UN and Appro_noRes have moderate resource violations at around 0.8. From Fig. 4 (c), we can see that algorithm Greedy has the minimum running time while ILP has the maximum running time.

We then investigated the performance of algorithms ILP, Appro, Appro_UN, Greedy, and Appro_noRes for the flow time minimization problem in terms of the average flow time, computing resource violation ratio, and running time, by varying the number of queries from 10 to 40 while fixing the maximum size of each dataset at 2,000 KB. As shown in Fig. 5 (a), the average flow time of all algorithms increases slightly with the growth on the number of queries. The reason is that a large number of query evaluation may make other being evaluated queries in later time slots, thereby increasing the average flow time. The resource violation ratio of computing resource keeps stable as depicted in Fig. 5 (b). On the other hand, the running times of different algorithms increase with the growth on the number of queries, as shown in Fig. 5 (c).

We thirdly studied the performance of different algorithms for the flow time minimization problem in terms of the average flow time, computing resource violation ratio, and running time, by varying the number of satellites in each orbit from 5 to 10. The results are shown in Fig. 6, from which it can be seen that the average flow time of queries decreases significantly when the number of satellites on each orbit increases from 5 to 7. The reason is that queries may need to wait for satellites moving into their transmission ranges when the number of satellites is low, pushing up query response delays. However, more satellites mean more available computing resource provided by the SEC network, which makes most queries being evaluated timely instead of waiting for being scheduled in later time slots. As shown in Fig. 6 (b), the resource violation ratio decreases with the growth on the number of satellites, since there are more computing resource available in the SEC network with more satellites. The running times of different algorithms decrease with the growth on the number of satellites, as shown in Fig. 6 (c). The rationale behind is that more time is required to wait for satellites moving into the range of ground users when the number of satellites is less than 7. However, the running times increase slightly afterwards, because more time is spent to find satellites for each ground user.

7.3 Performance of Different Algorithms for the Online Flow Time Minimization Problem

The rest is to evaluate the performance of algorithms OL, DOTS, CODE-V and Greedy_Ol for the online flow time minimization problem in terms of the average flow time, computing resource violation ratio, and running time, by varying the maximum size of each dataset from 2,000 KB to 6,000 KB. From Fig. 7 (a), we can see that the average flow time of those algorithms increases with the increase on the maximum size of a dataset. Also, the average flow time of algorithm OL is 16.3%, 19.5%, 13.1% lower than those of algorithms Greedy_Ol, CODE-V and DOTS, respectively. Furthermore, the performance gap between them is enlarging with the growth on the maximum size of each dataset. The reason is that algorithm OL leverages an online learning method to adaptively adjust the size of each data split [ds]. From Fig. 7 (b), we can see that the proposed algorithms OL and Greedy_Ol have moderate resource violation ratios at around 0.8. Note that algorithm DOTS and CODE-V have no computing resource violation ratios, due to the fact that both algorithms only select satellites with enough computing resource for each big data query. From Fig. 7 (c), we can see that algorithm DOTS has the maximum running time. The reason is that DOTS spent more time training DNN model. It must be mentioned our proposed online learning algorithm OL leverages a clever explore-and-exploit procedure with a bounded regret. It thus can start without connecting much data in real applications. This implicates efficiency and less workload for the runtime of distributed database management system for SEC networks. Besides, it can be adapted to new satellite applications, while DOTS requires time-consuming data collection, cleaning, and training.

We then investigated the performance of algorithms OL, DOTS, CODE-V and Greedy_Ol for the online flow time minimization problem in terms of the average flow time, computing resource violation ratio, and running time, by varying the number of satellites in each orbit from 5 to 10. First, Fig. 8 (a) shows the average flow times of algorithms OL, DOTS, CODE-V and Greedy_Ol. It can be seen that all the algorithms have almost similar flow times when the number of satellites grows from 5 to 6. However, the flow
The computing resource violation ratio.

Fig. 4. The performance of algorithms $\text{ILP}, \text{Appro}, \text{Appro_UN}, \text{Greedy}, \text{Appro_noRes}$ with different maximum sizes of datasets.

Fig. 5. The performance of algorithms $\text{ILP}, \text{Appro}, \text{Appro_UN}, \text{Greedy}, \text{Appro_noRes}$ with different numbers of queries.

Fig. 6. The performance of algorithms $\text{ILP}, \text{Appro}, \text{Appro_UN}, \text{Greedy}, \text{Appro_noRes}$ with different numbers of satellites in each orbit.

time of $\text{OL}$ is lower than that of other algorithms when the number of satellites increases from 7 to 10. The rationale behind is that when the number of satellites is low, the response delay is the major component of the flow time since ground users wait for satellites circulating into their transmission ranges. As the number of satellites grows from 7 to 10, algorithm $\text{OL}$ has a higher opportunity to find satellites that can evaluate queries in lower response, transmission and processing delays. Furthermore, algorithm $\text{DOTS}$ selects satellites in the communication range of the ground station to process computing tasks. Additionally, algorithm $\text{CODE-V}$ has a substantial probability of obtaining only local optimal task scheduling decision, since the mobility of satellites are not considered and long waiting delays can occur. Second, Fig. 8 (b) shows that the resource violation ratio decreases with the growth of the satellite number. The reason is that more satellites imply more computing resources available in the SEC network. The algorithms may have higher chances of assign queries to satellites with abundant resources. Third, Fig. 8 (c) shows that the running times of different resources decrease with the growth of satellite numbers. The rationale behind is that when there are fewer satellites in the SEC network, ground users need to spend more time waiting for satellites to enter the communication range of ground users. However, the running times of $\text{OL}$ and $\text{Greedy_{OL}}$ increase slightly afterwards, and the other two algorithms do not. This is because the running time of $\text{DOTS}$ is mainly used to train the DNN model and the running time of $\text{CODE-V}$ is mainly used to fitness evaluations and iterations.

We finally evaluated the performance of algorithms $\text{OL}$, $\text{DOTS}$, $\text{CODE-V}$ and $\text{Greedy_{OL}}$ in terms of the average flow time, computing resource violation ratio, and running time, by varying the number $\kappa$ of sub-intervals of range $Z$ from 4 to 24. We can see from Fig. 9 (a) that the average flow time of algorithm $\text{OL}$ is much lower than that of its counterparts. Also, the average flow time of $\text{OL}$ decreases if varying $\kappa$ from
The computing resource violation ratio.

Average flow time (s)

The implementation of the proposed algorithms for big data
manage the resource of the SEC network and schedule the
framework, such as Kubernetes or containers, can be used to
attached with an edge server or accelerator, virtualization
of satellite resources. Specifically, when each satellite can be
networking and cloud virtualization to enable orchestration
satellites can be adopted, which utilizes software defined
ment of the SEC network, the technique of software-defined
real SEC system requires real deploying of satellites in the
in the SEC network. Considering the implementation of a
analytics replies on the runtime that manages the resources
Future Implementation Directions

Z

shown in Fig. 9 (c), the running times of all algorithms also
and
Greedy_OL
do not split each dataset. The resource violation ratios of
OL and
Greedy_OL keep steady as shown in Fig. 9 (b). Also, as
shown in Fig. 9 (c), the running times of all algorithms also
keep steady as the impact of the number of sub-intervals of the
range \( Z \) on the running time of the algorithm is trivial.

7.4 Discussion on Integrating to Serverless Runtime
and Future Implementation Directions

The implementation of the proposed algorithms for big data
analytics replies on the runtime that manages the resources
in the SEC network. Considering the implementation of a
real SEC system requires real deploying of satellites in the
space and there lacks an open experimental platform, the real
implementation of an SEC network is not the major focus of
this paper. However, to enable efficient and effective manage-
ment of the SEC network, the technique of software-defined
satellites can be adopted, which utilizes software defined
networking and cloud virtualization to orchestration
of satellite resources. Specifically, when each satellite can be
attached with an edge server or accelerator, virtualization
framework, such as Kubernetes or containers, can be used to
manage the resource of the SEC network and schedule the
big data queries from ground users.

Recently, Function as a Service (FaaS) of serverless
computing paradigm is envisioned as a key technique to
provide agile, efficient, and high-performance runtime to
implementing various user requests in an SEC network.
Specifically, we can implement the big data query evaluation
as a service running in a serverless platform running upon
the SEC network. In particular, we assume that each satellite
is virtualized to run various serverless functions, and the
satellites in a network are managed via a software-defined
controller. The resources are managed via a management
and orchestration (MANO) module. On one hand, each big
data query can be divided into a number of sub-queries with
each being implemented in a serverless function. Or, the
evaluation of each dataset of a query can be implemented
as a serverless function. On the other hand, the proposed
algorithms can be implemented as a module, referred to as
a task scheduling module as shown in Fig. 10, to schedule
big data queries. Specifically, we have added a set of APIs
to the satellite simulation software proposed by existing
study [37], to obtain the available satellite resources and the
information of big data queries. However, considering the
limited satellite resources and the requirement for real-time
big data analysis, we may need to design low-overhead com-
munication protocols in SEC networks based on protocols
such as Message Queuing Telemetry Transport (MQTT) or
Constrained Application Protocol (CoAP) [25], for which we
consider as our future work.

8 Conclusion and Future Work

In this paper we formulated the flow time minimization
problem for query evaluation of big data analytics in an SEC
The computing resource violation ratio. We proposed an online learning algorithm with a bounded average flow time (s). We devise an approximation algorithm with an evaluation big data queries with various demands in such a minimization problem with uncertain volume of datasets, resource allocation, etc.; (3) the data plane uses virtualization and FaaS satellites and satellite-terrestrial link condition, topology change, and so on. Consequently, the controller is able to conduct topology control and resource allocation, etc.; (3) the data plane uses virtualization and FaaS technology to deploy edge computing platforms on each satellite. network, such that the average flow time of all queries is minimized, subject to the resource capacity on each satellite. For the offline version of the problem with a given set of queries, we devised an approximation algorithm with an approximation ratio, by reducing the problem to a minimum-cost multi-community flow problem. For the online flow time minimization problem with uncertain volume of datasets, we proposed an online learning algorithm with a bounded regret. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experiment results demonstrate that the proposed algorithms achieve at least 13% lower flow time than that of their comparison counterparts.

The future work built upon this study includes (1) there are different types of satellites in the space and air besides LEO satellites, such as high earth orbits satellites and unmanned aerial vehicles. Along with LEO satellites, such platforms form a hierarchical network with its computing nodes having different processing capabilities. How to evaluate big data queries with various demands in such a hierarchical satellite network is challenging; and (2) in some application scenarios, users on the ground may keep moving all the time to collect data from the environment. How to jointly consider the mobility of users and the dynamic availability of satellites is challenging, we will consider these as our next potential study in this topic.

References

Hao Wang is an associate professor at the Dalian University of Technology. Before that from 2016 to 2019, he was a researcher at GWDG and assistant lecturer at the University of Göttingen, Germany. After he received his PhD degree in Computer Science from the University of Kaiserlautern, Germany in 2015. His research interests include edge computing, network performance analysis, and next-generation networking.

Weifa Liang received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in Computer Science. He is a Professor in the Department of Computer Science at City University of Hong Kong. Prior to that, he was a Professor in the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing, network function virtualization, Internet of Things, software-defined networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He currently serves as an Associate Editor in the Editorial Board of IEEE Transactions on Communications, and is a senior member of IEEE.

Qiufen XIA received her PhD degree from the Australian National University in 2017, the ME degree and BSc degree from Dalian University of Technology in China in 2012 and 2009, all in Computer Science. She is currently an associate professor at the Dalian University of Technology. Her research interests include mobile cloud computing, query evaluation, big data analytics, big data management in distributed clouds, and cloud computing.

Shangguang Wang is a Professor at the School of Computer Science and Engineering, Beijing University of Posts and Telecommunications, China. He received his Ph.D. degree at Beijing University of Posts and Telecommunications in 2011. He has published more than 150 papers. His research interests include service computing, mobile edge computing, and satellite computing. He is currently serving as Chair of IEEE Technical Committee on Services Computing (2022-2013), and Vice-Chair of IEEE Technical Committee on Cloud Computing (2020-). He also served as General Chairs or Program Chairs of 10+ IEEE conferences. He is a Fellow of the IET, and Senior Member of the IEEE. For further information on Dr. Wang, please visit: http://www.sguangwang.com.