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 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 timelineness of streaming analytics in an SEC network. To address the afore-mentioned 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.**

I. Introduction

A. 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 tip of the iceberg of streaming analytics. With the growth of Internet-of-Thing (IoT) devices in remote areas, the overall volume of data streams dramatically increases over time. Since remote areas are usually without cellular network coverage, new computing paradigms that can enable streaming analytics are needed. Meanwhile, as the new space race keeps heating up, large satellite constellations in low-earth orbit (LEO) built by private companies are providing analytic services for applications in nearly any corner of the Earth [12], [22], [24], [48], [51], [61], [64]. In particular, by deploying computing resource to LEO satellites and interconnecting the satellites via inter-satellite links (ISLs), timely Big Data analytics can be performed at almost any corner of the Earth within the proximity of ground users. Report shows that the global satellite connection for data analytics of IoT applications in remote areas will more than double its revenue, going from $233 million in 2019 to $544 million in 2025 [30]. A fundamental problem of enabling such timely analytics is to evaluate analytical queries for Big Data, referred to as *Big Data analytical queries*, in a satellite edge computing (SEC) network while the datasets of the queries are distributed in dynamically moving satellites. An example of a Big Data query is to monitor surface temperature, vegetation coverage, or marine meteorological variations in a specific region for a given period [35] to infer the future trends of the natural environment in that region.

Manuscript received 14 January 2023; revised 29 October 2023; accepted 5 November 2023. Date of publication 13 November 2023; date of current version 27 November 2023. The work of Zichuan Xu and Qiufen Xia was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62172068 and 62172071, in part by the Natural Science Foundation of Liaoning under Grant 2023-MS-111 and the "Xinghai scholar" program. The work of Weifa Liang was supported by the City University of Hong Kong under Grants 9380137/CS and 7005845. Recommended for acceptance by Y. Yang. *(Corresponding author: Hao Wang.)*

Zichuan Xu, Guangyuan Xu, Hao Wang, 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 116621, China (e-mail: z.xu@dlut.edu.cn; 32017112@mail.dlut.edu.cn; haowang@dlut.edu.cn; qiufenxia@dlut.edu.cn).

Weifa Liang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: weifa.liang@cityu.edu.hk).

Shangguang Wang is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: sgwang@bupt.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPDS.2023.3332333, provided by the authors.

Digital Object Identifier 10.1109/TPDS.2023.3332333

B. 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 moves 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.

C. 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 timelineness 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 II summarizes related work. Section III introduces the system model, and defines the flow time minimization problem in an SEC network. Section IV presents an exact solution to the offline version of the flow time minimization problem. Section V devises an approximation algorithm for the flow-time minimization problem in an SEC network, assuming that all requests are given. Section VI proposes an online learning algorithm with a bounded regret for the online flow-time minimization problem in an SEC network. Section VII evaluates the performance of the proposed algorithms, and the paper is concluded in Section VIII.

II. Related Work

Big data analytics have gain much research attention by the research community. In the following, we summarize 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.

A. 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 anlytics 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.

B. 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.

C. 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.

III. Preliminaries

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

A. System Model

We consider a satellite edge computing (SEC) network $G = (\mathcal{S} \cup \mathcal{DC}, E)$, consisting of a set $\mathcal{S}$ of LEO satellites in the space and a set $\mathcal{DC}$ 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, inter-satellite 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.

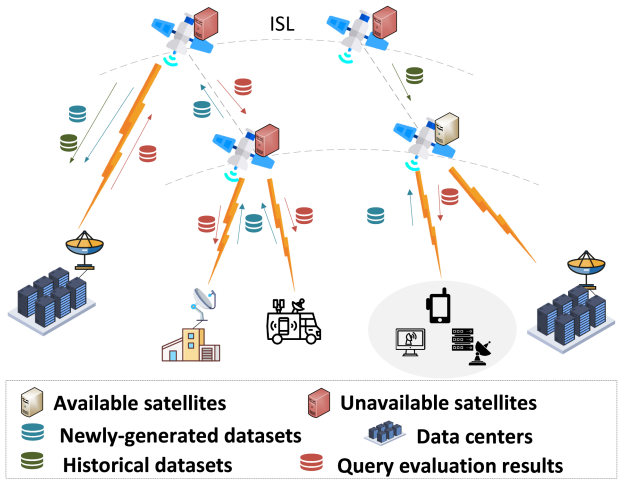


Fig. 1. Example of an SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$, provisioning query services for Big Data analytics.

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.

B. 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 τ_m within time horizon T. Let ds_m be the dataset that is generated by ground user u_i in time slot τ_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 $\mathcal{DS}_m$ 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 $r_m = \langle u_i, \tau_m, ds_m, \mathcal{DS}_m \rangle$.

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.

- *Case I:* Query r_m can be evaluated in an LEO satellite. The dataset ds_m that is newly generated by r_m needs to be uploaded to an LEO satellite. Besides, query r_m may require historical datasets in $\mathcal{DS}_m$ that were placed into the SEC network in previous time slots. As such, query r_m needs to be dispatched to satellites with its required datasets in $\mathcal{DS}_m$. In case the datasets in $\mathcal{DS}_m$ are not co-located in a single satellite, some of the datasets in $\mathcal{DS}_m$ have to be streamed to the location where the query is assigned for evaluation. Once being assigned to a location, the evaluation of query r_m can be evaluated immediately when the satellite receives the first unit D_{unit} of its dataset ds_m, which is referred to as *starting units*. This implies that the rest of its datasets can be uploaded continuously while the query is being evaluated. For example, an application of object detection can start processing a stream of images once it receives the first few images from IoT devices. In addition, while uploading data of a query, its destination satellite may move out of its transmission range. We thus assume that the data uploading and evaluation of a query execution can be handed over to other satellites in its range. The intermediate results of analyzing each dataset will be sent back to the user for final aggregation.
  Due to the resource availability of LEO satellites, the datasets required by each query r_m may be processed in a number of time slots after its arrival time slot τ_m. Its required datasets have to be forwarded to the selected satellites in its scheduled time slots. We thus use $y_{m,t,k}$ to indicate how much amount of a proportion of dataset $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ is processed in satellite s_k at time slot t.
- *Case II:* Query r_m can be evaluated in a data center in $\mathcal{DC}$ if no satellites have enough available computing resource. Considering that ground users are distributed in remote areas without the Internet access, we need to transmit the newly-generated datasets of the query to a satellite first via ground-air wireless communication links; the satellite then forward its received data to a data center afterwards. The historical datasets of the query will be directly downloaded from their satellites to a data center for the processing of the query. Denote by $y_{m,t,j}$ a decision variable indicating the amount of a proportion of dataset $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ is processed in data center DC_j of time slot t. Considering that $y_{m,t,j}$ can be any value in the range of $[0, |ds_m|]$, we assume that $y_{m,t,j}$ is determined according to the available Virtual Machines (VMs) of data centers and thus can be processed within the time slot t. However, the number of VMs that are leased to implement Big Data queries for a specific scenario usually is limited. We assume that a couple of VMs are leased from each public cloud.

C. Flow-Time of Big Data Query Evaluation

Once query r_m is scheduled for evaluation in the current time slot $t\,(\geq \tau_m)$, 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

$$C_m - \tau_m + 1, \tag{1}$$

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 that 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

$$x_m - \tau_m + 1. \tag{2}$$

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, then,

$$\eta_{m,k,t} = \frac{D_{unit}}{R_{i,t}} + \sum_{p_{m,k}} d_e \cdot D_{unit}, \tag{3}$$

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 $\mathcal{DS}_m$ that were already placed in the SEC network. For each dataset $ds_{m'} \in \mathcal{DS}_m$, let $\eta_{k,k'}$ be the delay of transferring a unit of dataset $ds_{m'}$ from satellite $s_{k'}$ to satellite s_k. Considering that each ISL link usually has abundant communication bandwidth, $\eta_{k,k'}$ is proportional to the amount of data required to be transferred via the path from $s_{k'}$ to s_k, i.e.,

$$\eta_{k,k'} = \sum_{e \in p_{k',k}} d_e \cdot D_{unit}, \tag{4}$$

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 ds_m needs to be forwarded from satellite s_k to data center $DC_j \in \mathcal{DC}$. 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 ds_m and the datasets in $\mathcal{DS}_m$ from satellites to data center DC_j. The delay of transmitting a unit amount of dataset $ds_{m'}$ from a satellite $s_{k'}$ to a data center DC_j can be calculated by (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 η_m^{proc} of evaluating query r_m in satellite s_k by

$$\begin{aligned}\eta_m^{proc} &= data\ size \times processing\ rate \\ &= \sum_{ds_{m'} \in \mathcal{DS}_m \cup \{ds_m\}} |ds_{m'}| \cdot \beta_k,\end{aligned} \tag{5}$$

where β_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 $DC_j \in \mathcal{DC}$ is

$$\eta_{m,j}^{proc} = \sum_{ds_{m'} \in \mathcal{DS}_m \cup \{ds_m\}} |ds_{m'}| \cdot \beta_j, \tag{6}$$

where β_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 responsed as soon as possible to make sure the results is timely, not necessarily in real-time.

D. 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 ξ 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 Δ be the length of each time slot t, if satellite s_k wants to process amounts $y_{m,t,k}$ of data for query r_m within $\Delta - (D_{unit}/R_{i,k})$ time, the amount of computing resource needed is

$$C(y_{m,t,k}) = \xi \cdot y_{m,t,k} \left(d/(\Delta - (D_{unit}/R_{i,k}))\right). \tag{7}$$

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

$$C(y_{m,t,k}) \leq CR_k. \tag{8}$$

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.

E. Problem Definitions

Given an SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$ with a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$ 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 τ_m to analyze a collection of datasets that consist of a newly-generated dataset ds_m and multiple historical datasets in $\mathcal{DS}_m$ 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 ds_m for r_m to a selected satellite s_k, transmitting the required historical datasets in $\mathcal{DS}_m$ 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 $s_k \in \mathcal{S}$. 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

TABLE I
LIST OF SYMBOLS

Symbols	Meaning
$G = (\mathcal{S} \cup \mathcal{DC}, E)$	An SEC network with a set $\mathcal{S}$ of satellites and a set $\mathcal{DC}$ of data centers.
s_k, DC_j, and u_i	A satellite in $\mathcal{S}$, a data center in $\mathcal{DC}$, and a ground user.
r_m and τ_m	A big data analytical query issued by u_i and its generated time slot.
ds_m and D_{unit}	The newly-generated dataset in τ_m issued by u_i, and the first unit of dataset ds_m
$\mathcal{DS}_m$	The set of datasets that are required for the evaluation of query r_m.
$y_{m,t,k}$	The amount of a proportion of dataset $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ is processed in satellite s_k of time slot t.
$y_{m,t,j}$	The amount of a proportion of dataset $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ is processed in data center DC_j of time slot t.
C_m	The time slot when the system completes the processing of big data analytical query r_m.
x_m	The time slot that query r_m is responded.
$R_{i,t}$	The achieved data rate via the wireless channel from ground user u_i to its closest satellite in time slot t.
$\eta_{m,k,t}$	The delay of uploading a unit of dataset of each query r_m to satellite s_k.
$p_{m,k}$	The path from query r_m's closest satellite in its transmission range to the selected satellite s_k.
η_m^{proc} and Δ	The delay of evaluating query r_m in satellite s_k, and the length of each time slot t.
CR_k and CR_j	The capacities of computing resources on satellite s_k and a virtual machine in DC_j.
$x_{m,t}$	The binary indicator variables that show whether query r_m is scheduled in a satellite.
$x'_{m,t}$	The binary indicator variables that show whether query r_m is scheduled in a data center.
ξ and d	The amount ξ of computing resource in time duration d that can process a unit amount of the datasets of queries.
L_o	The potential location for each query and $L_o \in \{\mathcal{S} \cup \mathcal{DC}\}$ with $1 \le o \le \lvert\mathcal{S} \cup \mathcal{DC}\rvert$.
$\lvert ds\rvert$ and $\lvert f_{k,t}\rvert$	The size of each data split, and the number of datasets that are routed in each time widget of time slot t.
f and $c(f)$	The flow found in approximation algorithm and the total cost of flow f.
Φ	The total flow time derived from $c(f)$.
$\lvert ds_{max}\rvert$ and $\lvert ds_{min}\rvert$	The maximum and minimum sizes of each dataset can be obtained according to historical information.
Z and κ	The range of $\lvert ds\rvert$ and $Z = [\lvert ds_{min}\rvert, \lvert ds_{max}\rvert]$ and the number of subintervals in the range of Z.
ε	The length of each subinterval in the range of Z.

newly-generated dataset ds_m of r_m to a selected satellite s_k, transmitting its historical datasets in $\mathcal{DS}_m$ 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 $s_k \in \mathcal{S}$.

For clarity, the symbols used in this paper are summarized in Table I.

IV. 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 $\mathcal{S}_{m,t}$ 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

$$\textbf{ILP}: \min \left(\sum_{r_m \in R} (C_m - \tau_m + 1) \right) / |R|. \tag{9}$$

Recall that $y_{m,t,k}$ indicates the amount of a proportion of dataset $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ 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 $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ 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 $ds_{\max} = \max\{ds_m\}$, the constraints of the ILP formulation is given as follows.

$$\sum_{t \ge \tau_m} \sum_{k,j} (y_{m,t,k} + y_{m,t,j}) = |ds_m|, \forall r_m \tag{10}$$

$$\sum_{t \ge \tau_m} \sum_{k,j} (y_{m',t,k} + y_{m',t,j}) = |ds_{m'}|, \forall r_m, ds_{m'} \in \mathcal{DS}_m \tag{11}$$

$$\sum_{r_m \in R} \sum_{t \ge \tau_m \&\& s_k \in \mathcal{S}_{m,t}} \frac{\xi \cdot y_{m,t,k} \cdot d}{(\Delta - \eta_{m,k,t})} \le CR_k, \forall t, s_k \tag{12}$$

$$\sum_{r_m \in R} \sum_{t \ge \tau_m} \frac{\xi \cdot y_{m,t,j} \cdot d}{(\Delta - \eta_{m,k,t} - \eta_{k,j})} \le CR_j, \forall t, DC_j \in \mathcal{DC} \tag{13}$$

$$x_{m,t} \ge (y_{m,t,k} + y_{m,t,j}) / |ds_{\max}|, \forall r_m \, and \, \forall t \tag{14}$$

$$x'_{m,t} \ge (y_{m',t,k} + y_{m',t,j}) / |ds_{\max}|, \forall r_m, \forall t, \forall ds_{m'} \in DS_m \tag{15}$$

$$(x_{m,t} + x'_{m,t}) \in \{0, 1\} \tag{16}$$

$$x_{m,t}, x'_{m,t} \in \{0, 1\} \tag{17}$$

$$y_{m,t,k}, y_{m,t,j} \in [1, |ds_m|]^+. \tag{18}$$

$$y_{m',t,k}, y_{m',t,j} \in [1, |ds_{m'}|]^+. \tag{19}$$

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 τ_m. Constraint (11) ensures that the historical datasets in DS_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 (12) includes the time spent for uploading a unit amount of data D_{unit} of r_m to enable subsequent data processing at time slot t. Intuitively, if the transmission of a unit amount of data D_{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 Δ 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 1 s if there are assigned data of r_m for processing at time slot 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.

V. 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.

A. 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.

B. 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.

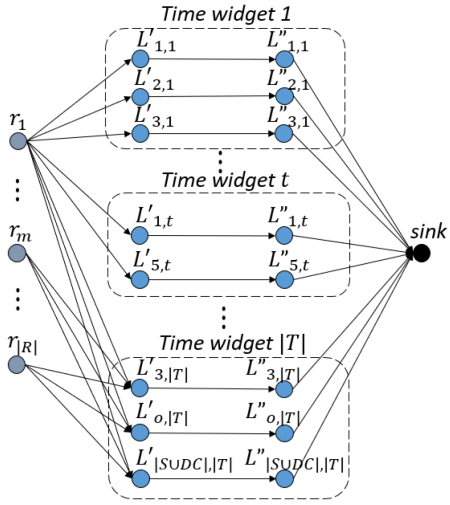


Fig. 2. Example of the auxiliary graph $G' = (V', E')$, where a query r_m is only connected to a time widget if the time of the widget is later than or equal to the generation time τ_m of r_m.

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 τ_m. The resource availability of satellites and data centers in time slots no earlier than τ_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 $L_o \in \{\mathcal{S} \cup \mathcal{DC}\}$ with $1 \leq o \leq |\mathcal{S} \cup \mathcal{DC}|$.

To incorporate such dynamics, we create $2 \cdot T$ copies for each location in $\{\mathcal{S} \cup \mathcal{DC}\}$, which are referred to as *virtual locations*. We then add all the virtual locations of each $L_o \in \{\mathcal{S} \cup \mathcal{DC}\}$ 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., $E' \leftarrow E' \cup \{\langle r_m, L'_{o,t}\rangle\}$. The rationale behind is that the query r_m can only be evaluated after its generation at time slot τ_m. The capacity of each of such edges is set to

$$u(\langle r_m, L'_{o,t}\rangle) = |\mathcal{DS}_m| + 1, \tag{20}$$

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

$$c(r_m, L'_{o,t}) = t - \tau_m + 1. \tag{21}$$

We then connect the virtual locations in each time widget. Specifically, there is a directed edge $\langle L'_{o,t}, L''_{o,t}\rangle$ from virtual

Algorithm 1: An Approximation Algorithm for the Flow-Time Minimization Problem for Big Data Analytics With a Uniform Data Size in an SEC Network, i.e., `ApproUN`.

Input: An SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$, a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$ 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 τ_m to analyze its required set of datasets with a uniform data size.

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

1: /* **Stage 1**: constructing an auxiliary graph G'*/
2: Construct an auxiliary graph as illustrated in Fig. 2, by adding a number of T time widgets with each having virtual locations for query evaluation, query nodes into V' and connecting the nodes in V';
3: /* **Stage 2**: finding a minimum cost flow in the auxiliary graph G' and transferring the obtained flow into a feasible solution*/
4: Consider each query r_m as a commodity with a demand of $|\mathcal{DS}_m| + 1$ that needs to be routed from query node r_m to the $sink$ in G';
5: Find a minimum cost multicommodity flow in auxiliary graph G' following the approximation algorithm in [13];
6: The flow in each edge $\langle L'_{o,t}, L''_{o,t}\rangle$ is considered the datasets that are assigned to location L_o to evaluate query r_m;
7: **return** The scheduling and resource allocation for each r_m;

location $L'_{o,t}$ to virtual location $L''_{o,t}$, 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

$$u(L'_{o,t}, L''_{o,t}) = \left\lfloor (CR_k(\Delta - \eta_{\min}))/(\xi \cdot d \cdot |ds|) \right\rfloor, \quad (22)$$

where $\eta_{\min} = \min\{\min_{m,k,t} \eta_{m,k,t}, \min_{k,j} \eta_{k,j}\}$. The cost of edge $\langle L'_{o,t}, L''_{o,t}\rangle$ 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., $E' \leftarrow E' \cup \{\langle L''_{o,t}, sink\rangle\}$, 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 *a commodity* with demand $|\mathcal{DS}_m| + 1$. Let f be the flow found. The fractional flow of f routed via edge $\langle L'_{o,t}, L''_{o,t}\rangle$ represents the datasets to be assigned to location L_o for processing. The detailed approximation algorithm is given in *Algorithm* 1, which is referred to as `ApproUN` for short.

Algorithm 2: An Approximation Algorithm for the Flow-Time Minimization Problem for Big Data Analytics in an SEC Network, i.e., `Appro`.

Input: An SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$, a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$ 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 τ_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 $ds_m \in \mathcal{DS}_m \cup \{ds_m\}$ of query r_m into $\lceil \frac{|ds_m|}{|ds|} \rceil$ 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;

C. 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 $\lceil \frac{|ds_m|}{|ds|} \rceil$ data splits. Given the 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* `Appro`.

D. Algorithm Analysis

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

Lemma 1: The solution delivered by *Algorithm* `ApproUN` violates the computing resource capacity of each satellite by a maximum ratio of $\frac{1-1/\alpha}{1-1/\beta}$ in the worst case, where

$\alpha = \frac{\Delta}{\eta_{\min}}$, $\beta = \frac{\Delta}{\eta_{\max}}$, $\eta_{\max} = \{\max_{m,k,t} \eta_{m,k,t}, \max_{k,j} \eta_{k,j}\}$, $\eta_{\min} = \{\min_{m,k,t} \eta_{m,k,t}, \min_{k,j} \eta_{k,j}\}$.

Please see Appendix, available online, for the detailed proof.

Theorem 1: Given an SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$ with a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$ 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 τ_m to analyze its required set of datasets consisting of both a newly-generated dataset ds_m and previously-generated datasets in $\mathcal{DS}_m$ 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, available online, for the detailed proof.

Theorem 2: Given an SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$ with a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$ 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 τ_m to analyze its required set of datasets consisting of both a newly-generated dataset ds_m and previously-generated datasets in $\mathcal{DS}_m$ that are already placed in the SEC network. *Algorithm* `Appro` 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 $\frac{1-1/\alpha}{1-1/\beta} \lfloor \frac{|ds_{\max}|}{|ds|} \rfloor$, where T is the length of a finite time horizon.

Please see Appendix, available online, for the detailed proof.

VI. 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.

A. 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 $|ds|$ 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 and schedule queries to satellites adaptively. Specifically, we assume that the maximum and minimum sizes of each dataset can be obtained according to historical information, which are denoted by $|ds_{\max}|$ and $|ds_{\min}|$, respectively. We discretize the value of $|ds|$ according to its maximum and minimum values. The design rationale of the proposed online algorithm is to leverage a customized Lipschitz bandit learning method for $|ds|$ among the discretized range of $|ds|$ that having a finite number of values. After obtaining a suitable value for $|ds|$, we invoke *Algorithm* `Appro` in each time slot. An example of the basic idea of the proposed online learning algorithm is shown in Fig. 3.

B. Online Learning Algorithm

Finding an appropriate value of $|ds|$ is challenging, since there are infinite numbers of such values in the range of $Z = [|ds_{\min}|, |ds_{\max}|]$. 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.

$$|R(a) - R(b)| \leq \psi |a - b|, \text{ for any arms } a, b \in Z, \quad (23)$$

where ψ is a given constant.

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

Theorem 3: Given an SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$ with a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$ 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 τ_m to analyze its required set of datasets consisting of both a newly-generated dataset ds_m and previously-generated datasets in $\mathcal{DS}_m$ that are already placed

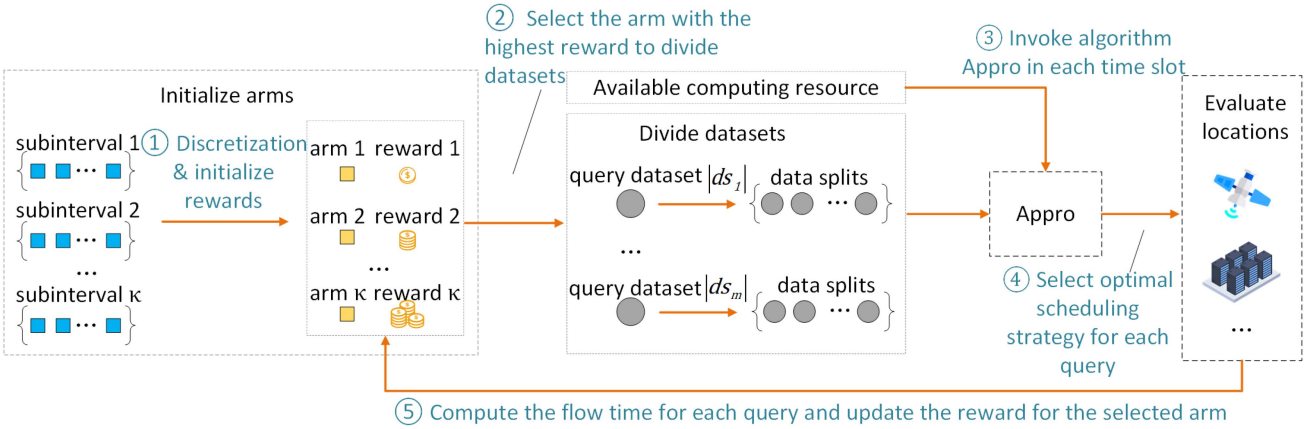


Fig. 3. Example of the basic idea of algorithm OL.

Algorithm 3: An Online Learning Algorithm for the Online Flow-Time Minimization Problem for Big Data Analytics in an SEC Network, i.e., OL.

Input: An SEC network $G = (\mathcal{S} \cup \mathcal{DC}, E)$, a set $\mathcal{S}$ of LEO satellites and a set $\mathcal{DC}$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 τ_m to analyze its required set of datasets consisting of both a newly-generated dataset ds_m and previously-generated datasets in $\mathcal{DS}_m$ that are already placed in the SEC network.

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

1: Divide the interval Z into κ intervals with fixed length $\varepsilon = \frac{|ds_{\max}|-|ds_{\min}|}{\kappa}$, and let Z' be the discretized set of arms;
2: **for** each time slot $t \leftarrow 1 \cdots T$ **do**
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 $|ds|$;

in the SEC network. Assuming that the queries arrive into the system dynamically without the knowledge of future arrivals, there is an online learning algorithm, *Algorithm* OL, for the online flow-time minimization problem in G, with the bounded regret $O(\sqrt{\kappa T \log T} + T \cdot \eta \cdot \varepsilon)$, where κ is the number of intervals when the range Z is discretized, $\varepsilon = \frac{|ds|_{\max}-|ds_{\min}|}{\kappa}$, T is the time horizon, and ψ is a given constant of the Lipschitz condition in (23).

Please see Appendix, available online, for the detailed proof.

VII. 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.

A. 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 light-weight 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 κ 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 drawn from $[1,000, 6,000]$ Kilo Bytes (KB) [7], [32], [44], [62]. The number of datasets required by each query is randomly withdrawn 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.

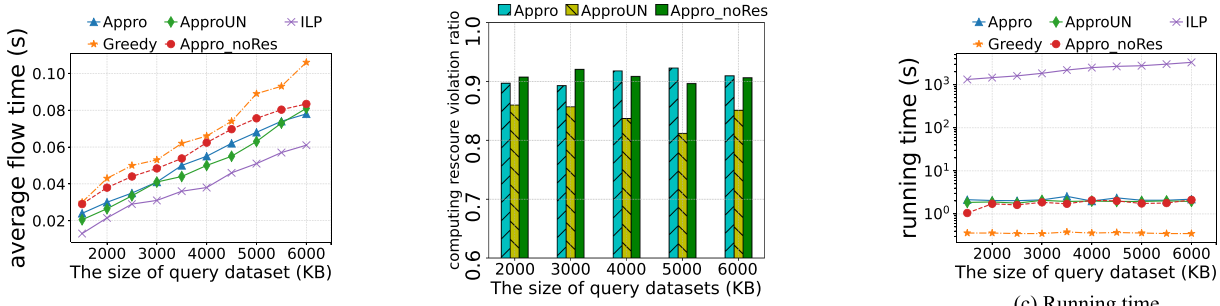
The metrics adopted in evaluating the proposed algorithms against their benchmarks include the average flow time, the resource violation ratio, and the running time of algorithms. The optimization objectives of the flow time minimization problem and the online flow time minimization problem are to minimize the flow time while guaranteeing the resource constraints. Note that algorithm ILP 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 ILP. Similarly, algorithm Greedy does not violate the resource capacities on satellites as it selects from the satellites with sufficient available resources. However, algorithm Greedy_OL evaluates queries according to the resource availabilities in its arriving time slot. This may violate the resource capacity on a satellite, because the resource availability of each satellite dynamically changes and if it does not have enough resources in future time slots. Without otherwise specified, the obtained values of these metrics are the average of 15 runs of the algorithms.

B. 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 than that of algorithm Appro, because Appro_UN deals with a special case of the flow time minimization problem with all datasets having a 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.

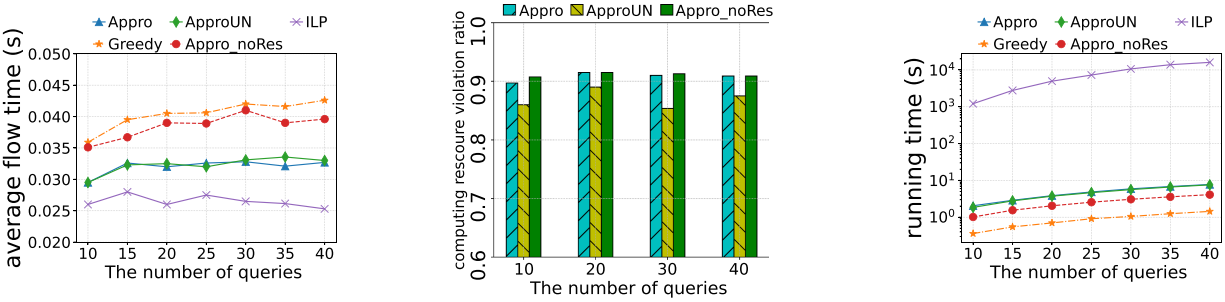
We then investigated the performance of algorithms ILP, Appro, Appro_UN, Greedy, and Appro_noRes for the flow time minimization problem in terms of the average flow time, computing resource violation ratio, and running times, by varying the number of queries from 10 to 40 while fixing the maximum size 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 third 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 times, 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



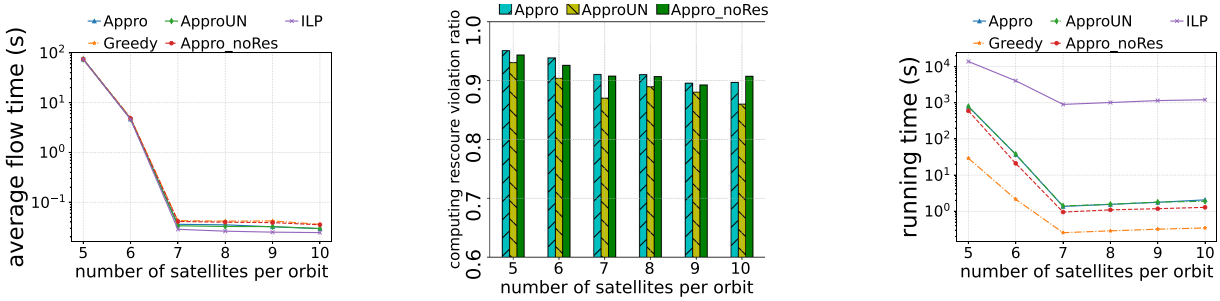
(a) The average flow time. (b) The computing resource violation ratio. (c) Running time.

Fig. 4. Performance of algorithms ILP, Appro, Appro_UN, Greedy, Appro_noRes with different maximum sizes of datasets.



(a) The average flow time. (b) The computing resource violation ratio. (c) Running time.

Fig. 5. Performance of algorithms ILP, Appro, Appro_UN, Greedy, Appro_noRes with different numbers of queries.



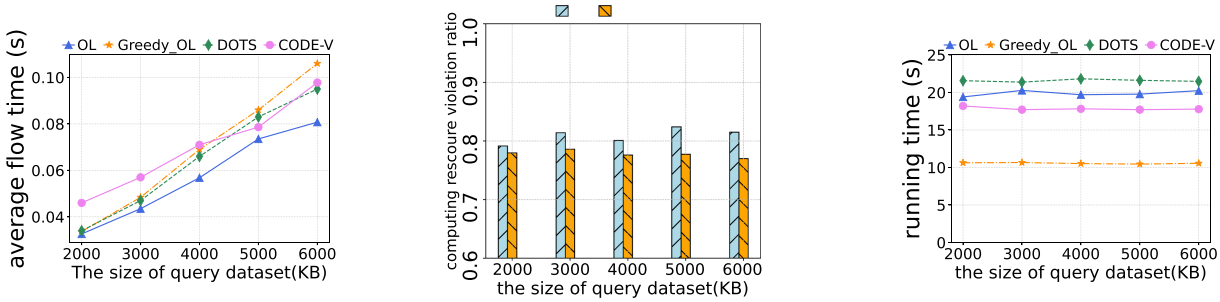
(a) The average flow time. (b) The computing resource violation ratio. (c) Running time.

Fig. 6. Performance of algorithms ILP, Appro, Appro_UN, Greedy, Appro_noRes with different numbers of satellites in each orbit.

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.

C. *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%,

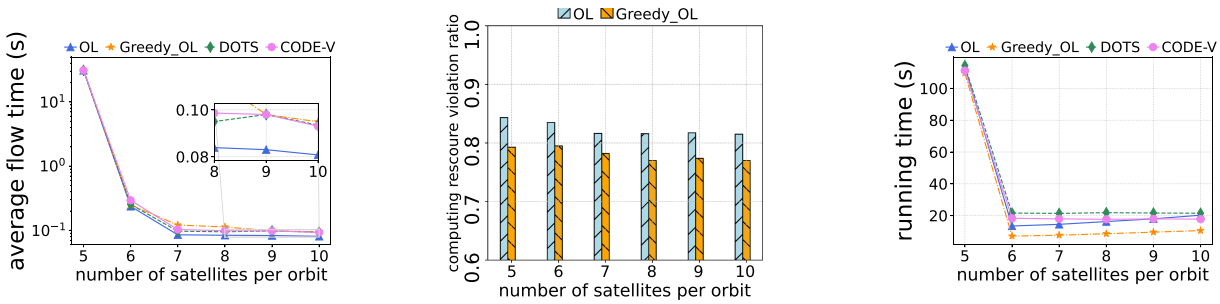


(a) The average flow time.

(b) The computing resource violation ratio.

(c) Running time.

Fig. 7. Performance of algorithms OL, Greedy_OL, CODE-V and DOTS with different maximum sizes of data sets.



(a) The average flow time.

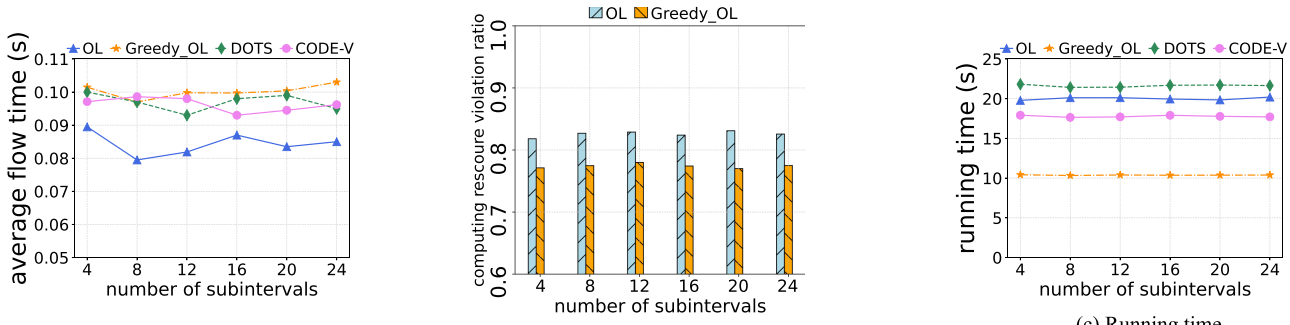
(b) The computing resource violation ratio.

(c) Running time.

Fig. 8. Performance of algorithms OL, Greedy_OL, CODE-V and DOTS with different numbers of satellites in each orbit.

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 time of 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 OL has a higher opportunity to find satellites that can evaluate queries in lower response, transmission and processing delays. Furthermore, algorithm DOTS selects satellites in the communication range of the ground station to process computing tasks. Additionally, algorithm 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 algorithms 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 OL and Greedy_OL increase slightly afterwards, and the other two algorithms do not. This is



(a) The average flow time.

(b) The computing resource violation ratio.

(c) Running time.

Fig. 9. Performance of algorithms OL, Greedy_OL, CODE-V and DOTS with different numbers κ of sub-intervals of range Z.

because the running time of DOTS is mainly used to train the DNN model and the running time of CODE-V is mainly used to fitness evaluations and iterations.

We finally evaluated the performance of algorithms OL, DOTS, CODE-V and Greedy_OL in terms of the average flow time, computing resource violation ratio, and running time, by varying the number κ of sub-intervals of range Z from 4 to 24. We can see from Fig. 9(a) that the average flow time of algorithm OL is much lower than that of its counterparts. Also, the average flow time of OL decreases if varying κ from 4 to 8, and increases afterwards. This is because when κ is small, the data split $|ds|$ is large, leading to large resource slices that cannot be utilized in satellites and forcing most queries to be evaluated in data centers. Further, when κ is large, the data split is small. Algorithm OL spends too much time to find a suitable size for each data split, which leads to a higher flow time in the end. In addition, the number κ of sub-intervals has almost no impact on the average flow time of DOTS and CODE-V, because DOTS and CODE-V 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.

D. 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 management of the SEC network, the technique of software-defined satellites can be adopted, which utilizes software defined networking and cloud virtualiztion to enable 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 communication 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.

VIII. Conclusion and Future Work

In this paper we formulated the flow time minimization problem for query evaluation of Big Data analytics in an SEC 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.

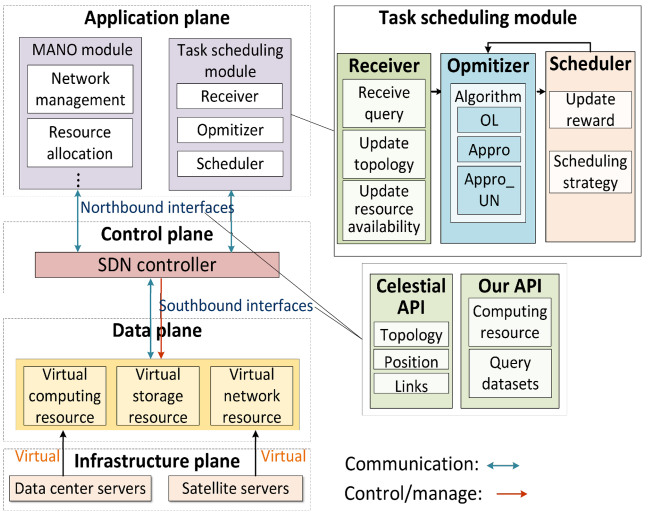


Fig. 10. Architecture for SEC networks: (1) application plane: includes MANO module and task scheduling module. We integrate proposed algorithms into the task scheduling module to obtain the optimal scheduling and resource allocation strategy; (2) the control plane includes an SDN controller. By collecting and monitoring the network status, an SDN controller can quickly obtain information including resource availability, 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.

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

[1] L. Boero, R. Bruschi, F. Davoli, M. Marchese, and F. Patrone, "Satellite networking integration in the 5G ecosystem: Research trends and open challenges," *IEEE Netw.*, vol. 32, no. 5, pp. 9–15, Sep./Oct. 2018.

[2] A. Boukerche and V. Soto, "An efficient mobility-oriented retrieval protocol for computation offloading in vehicular edge multi-access network," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2675–2688, Jun. 2020.

[3] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu, "iRAF: A deep reinforcement learning approach for collaborative mobile edge computing IoT networks," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7011–7024, Aug. 2019.

[4] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[5] N. Cheng et al., "Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[6] Y. C. Chou, X. Ma, F. Wang, S. Ma, S. H. Wong, and J. Liu, "Towards sustainable multi-tier space networking for LEO satellite constellations," in *Proc. IEEE/ACM 30th Int. Symp. Qual. Serv.*, 2022, pp. 1–11.

[7] X. Cheng, W. Tian, F. Shi, M. Zhao, S. Chen, and H. Wang, "A blockchain-empowered cluster-based federated learning model for blade icing estimation on IoT-enabled wind turbine," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9184–9195, Dec. 2022.

[8] L. Chen, F. Tang, and X. Li, "Mobility- and load-adaptive controller placement and assignment in LEO satellite networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[9] L. Chen, F. Tang, Z. Li, L. T. Yang, J. Yu, and B. Yao, "Time-varying resource graph based resource model for space-terrestrial integrated networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[10] C. Ding, J.-B. Wang, H. Zhang, M. Lin, and G. Y. Li, "Joint optimization of transmission and computation resources for satellite and high altitude platform assisted edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 1362–1377, Feb. 2022.

[11] R. Deng, B. Di, S. Chen, S. Sun, and L. Song, "Ultra-dense LEO satellite offloading for terrestrial networks: How much to pay the satellite operator?," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6240–6254, Oct. 2020.

[12] S. Fu, J. Gao, and L. Zhao, "Collaborative multi-resource allocation in terrestrial-satellite network towards 6G," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7057–7071, Nov. 2021.

[13] A. V. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein, "An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow," in *Proc. Int. Conf. Integer Program. Combinatorial Optim.*, Springer, 1998, pp. 338–352.

[14] M. Guan, T. Xu, F. Gao, W. Nie, and H. Yang, "Optimal walker constellation design of LEO-based global navigation and augmentation system," *Remote Sens.*, vol. 12, no. 11, 2020, Art. no. 1845.

[15] H. Huang, S. Guo, W. Liang, K. Wang, and Y. Okabe, "Coflow-like online data acquisition from low-earth-orbit datacenters," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2743–2760, Dec. 2020.

[16] H. Huang, S. Guo, W. Liang, K. Wang, and A. Y. Zomaya, "Green data-collection from geo-distributed IoT networks through low-earth-orbit satellites," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 806–816, Sep. 2019.

[17] Iridium. Accessed: Jul. 2023. [Online]. Available: https://www.iridium.com/network/

[18] X. Jia, T. Lv, F. He, and H. Huang, "Collaborative data downloading by using inter-satellite links in LEO satellite networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1523–1532, Mar. 2017.

[19] H. Jeong, H. Lee, C. Shin, and S. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. ACM Symp. Cloud Comput.*, 2018, pp. 401–411.

[20] Y. Jin, Z. Qian, S. Guo, S. Zhang, L. Jiao, and S. Lu, "runData: Redistributing data via piggybacking for geo-distributed data analytics over edges," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 40–55, Jan. 2022.

[21] Z. Jia, M. Sheng, J. Li, D. Zhou, and Z. Han, "VNF-based service provision in software defined LEO satellite networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6139–6153, Sep. 2021.

[22] Z. Jia, M. Sheng, J. Li, D. Zhou, and Z. Han, "Joint HAP access and LEO satellite backhaul in 6G: Matching game-based approaches," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1147–1159, Apr. 2021.

[23] T. Kim, J. Kwak, and J. P. Choi, "Satellite edge computing architecture and network slice scheduling for IoT support," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14938–14951, Aug. 2022, doi: 10.1109/JIOT.2021.3132171.

[24] C. Liu, W. Feng, Y. Chen, C. -X. Wang, and N. Ge, "Cell-free satellite-UAV networks for 6G wide-area Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1116–1131, Apr. 2021.

[25] M. Luglio, M. Marchese, F. Patrone, C. Roseti, and F. Zampognaro, "Performance evaluation of a satellite communication-based MEC architecture for IoT applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 5, pp. 3775–3785, Oct. 2022.

[26] X. Li et al., "Processing-while-transmitting: Cost-minimized transmission in SDN-Based STINs," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 243–256, Feb. 2022.

[27] Q. Li et al., "Service coverage for satellite edge computing," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 695–705, Jan. 2022.

[28] Q. Li, S. Wang, X. Ma, A. Zhou, and F. Yang, "Towards sustainable satellite edge computing," in *Proc. IEEE Int. Conf. Edge Comput.*, 2021, pp. 1–8.

[29] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[30] S. Lucero, *Satellite IoT market report 2020. Omdia*. London, U.K., White Paper, Mar. 2020.

[31] Md. M. Hussain and M. M. Sufyan Beg, "CODE-V: Multi-hop computation offloading in vehicular fog computing," *Future Gener. Comput. Syst.*, vol. 116, pp. 86–102, 2021.

[32] P. K. Malik et al., "Industrial Internet of Things and its applications in industry 4.0: State of the art," *Comput. Commun.*, vol. 166, pp. 125–139, 2021.
[33] NVIDIA. Accessed: Jul. 2023. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/
[34] A. Ndikumana et al., "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.
[35] Northern Sky Research, "Big data analytics via satellite." Accessed: Jul. 2023. [Online]. Available: https://www.nsr.com/?research=big-data-analytics-via-satellite-5th-edition
[36] M. Orabi, J. Khalife, and Z. M. Kassas, "Opportunistic navigation with doppler measurements from Iridium next and Orbcomm LEO satellites," in *Proc. IEEE Aerosp. Conf.*, 2021, pp. 1–9.
[37] T. Pfandzelter and D. Bermbach, "Celestial: Virtual software system testbeds for the LEO edge," in *Proc. 23rd Int. Middleware Conf.*, 2022, pp. 69–81.
[38] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh, "Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view," *IEEE Commun. Surv. Tuts.*, vol. 18, no. 4, pp. 2442–2473, Fourth Quarter 2016.
[39] SpaceX FCC update, "SpaceXnon-geostationary satellite system," 2018. [Online]. Available: https://www.ic.gc.ca/eic/site/smt-gst.nsf/vwapj/SLPB-005--18-SpaceX-attachment2.pdf
[40] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "DRL-based V2V computation offloading for blockchain-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3882–3897, Jul. 2023.
[41] Z. Song, Y. Hao, Y. Liu, and X. Sun, "Energy-efficient multiaccess edge computing for terrestrial-satellite Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14202–14218, Sep. 2021.
[42] Y. Su, Y. Liu, Y. Zhou, J. Yuan, H. Cao, and J. Shi, "Broadband LEO satellite communications: Architectures and key technologies," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 55–61, Apr. 2019.
[43] F. Tang, "Dynamically adaptive cooperation transmission among satellite-ground integrated networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1559–1568.
[44] Q. Tang, Z. Fei, B. Li, and Z. Han, "Computation offloading in LEO satellite networks with hybrid cloud and edge computing," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9164–9176, Jun. 2021.
[45] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.
[46] F. Tang, H. Zhang, and L. T. Yang, "Multipath cooperative routing with efficient acknowledgement for LEO satellite networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 179–192, Jan. 2019.
[47] G. Pan, J. Ye, J. An, and S. Alouin, "Latency versus reliability in LEO mega-constellations: Terrestrial, aerial, or space relay?," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5330–5345, Sep. 2023, doi: 10.1109/TMC.2022.3168081.
[48] Q. -V. Pham et al., "Aerial computing: A new computing paradigm, applications, and challenges," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8339–8363, Jun. 2022, doi: 10.1109/JIOT.2022.3160691.
[49] F. Vatalaro, G. E. Corazza, C. Caini, and C. Ferrarelli, "Analysis of LEO, MEO, and GEO global mobile satellite systems in the presence of interference and fading," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 2, pp. 291–300, Feb. 1995.
[50] C. Wang, C. Chen, Q. Pei, Z. Jiang, and S. Xu, "An information-centric in-network caching scheme for 5G-enabled internet of connected vehicles," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3137–3150, Jun. 2023.
[51] Y. Wang, W. Feng, J. Wang, and T. Q. S. Quek, "Hybrid satellite-UAV-terrestrial networks for 6G ubiquitous coverage: A maritime communications perspective," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3475–3490, Nov. 2021.
[52] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.
[53] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 2, pp. 598–611, Feb. 2022.
[54] Q. Xia, W. Ren, M. Li, and J. Ren, "Age-aware query evaluation for big data analytics in mobile edge clouds," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun. IEEE 18th Int. Conf. Smart City IEEE 6th Int. Conf. Data Sci. Syst.*, 2020, pp. 214–222.
[55] Q. Xia, L. Zhou, W. Ren, and Y. Wang, "Proactive and intelligent evaluation of big data queries in edge clouds with materialized views," *Comput. Netw.*, vol. 203, 2022, Art. no. 108664.
[56] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Netw.*, vol. 34, no. 3, pp. 224–231, May/Jun. 2020.
[57] Z. Xu et al., "Energy-aware inference offloading for DNN-driven applications in mobile edge clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 4, pp. 799–814, Apr. 2021.
[58] Z. Xu et al., "Online learning algorithms for offloading augmented reality requests with uncertain demands in MECs," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 1064–1074.
[59] X. Xu et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, 2019.
[60] S. Yu, X. Gong, Q. Shi, X. Wang, and X. Chen, "EC-SAGINs: Edge-computing-enhanced space–air–ground-integrated networks for internet of vehicles," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5742–5754, Apr. 2022.
[61] L. Yan, X. Fang, L. Hao, and Y. Fang, "Safety-oriented resource allocation for space-ground integrated cloud networks of high-speed railways," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2747–2759, Dec. 2020.
[62] W. Yu, Y. Liu, T. Dillon, W. Rahayu, and F. Mostafa, "An integrated framework for health state monitoring in a smart factory employing IoT and big data techniques," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2443–2454, Feb. 2022.
[63] Q. Yang, X. Luo, P. Li, T. Miyazaki, and X. Wang, "Computation offloading for fast CNN inference in edge computing," in *Proc. Conf. Res. Adaptive Convergent Syst.*, 2019, pp. 101–106.
[64] L. Zhen, A. K. Bashir, K. Yu, Y. D. Al-Otaibi, C. H. Foh, and P. Xiao, "Energy-efficient random access for LEO satellite-assisted 6G Internet of remote things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5114–5128, Apr. 2021.
[65] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.
[66] C. Zhou et al., "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 911–925, Feb. 2021.
[67] Z. Zhou, H. Yu, C. Yu, Z. Chang, S. Mumtaz, and J. Rodriguez, "BEGIN: Big data enabled energy-efficient vehicular edge computing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 82–89, Dec. 2018.
[68] X. Zhang, Y. Yang, M. Xu, and J. Luo, "ASER: Scalable distributed routing protocol for LEO satellite networks," in *Proc. IEEE 46th Conf. Local Comput. Netw.*, 2021, pp. 65–72.
[69] Z. Zhang, W. Zhang, and F. H. Tseng, "Satellite mobile edge computing: Improving QoS of high-speed satellite-terrestrial networks using edge computing techniques," *IEEE Netw.*, vol. 33, no. 1, pp. 70–76, Jan./Feb. 2019.

Zichuan Xu (Member, IEEE) received the BSc and ME degrees in computer science from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the PhD degree in computer science from the Australian National University, in 2016. From 2016 to 2017, he was a research associate with the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently a full professor and PhD advisor with the School of Software, Dalian University of Technology. He is also a 'Xinghai Scholar' in Dalian University of Technology. His research interests include edge computing, cloud computing, serverless computing, network function virtualization, algorithmic game theory, and optimization problems.

Guangyuan Xu received the BE degree from the North China University of Science and Technology, China, in 2020. He is currently working toward the PhD degree with the School of Software, Dalian University of Technology, Dalian, China. His current research interests include Big Data analytics, satellite edge computing, and resource allocation.

Hao Wang (Member, IEEE) received the PhD degree in computer science from the University of Kaiserslautern, Germany, in 2015. He is an associate professor with the Dalian University of Technology. Before that from 2016 to 2019, he was a researcher with GWDG and assistant lecturer with the University of Göttingen, Germany. His research interests include edge computing, network performance analysis, and next-generation networking.

Weifa Liang (Senior Member, IEEE) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China, in 1989, and the PhD degree from the Australian National University, in 1998, all in computer science. He is a professor with the Department of Computer Science, City University of Hong Kong. Prior to that, he was a professor with 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*.

Qiufen Xia (Member, IEEE) received the BSc and ME degrees in computer science from the Dalian University of Technology, China, in 2009 and 2012, respectively, and the PhD degree in computer science from the Australian National University, in 2017. She is currently an associate professor with 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 (Senior Member, IEEE) received the PhD degree from the Beijing University of Posts and Telecommunications, in 2011. He is a professor with the School of Computer Science and Engineering, Beijing University of Posts and Telecommunications, China. 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 more than 10 IEEE conferences. He is a fellow of the IET. For further information on him, please visit: http://www.sguangwang.com.