INFOCOM 2024 1570929085

In-Orbit Processing or Not? Sunlight-Aware Task Scheduling for Energy-Efficient Space Edge Computing Networks

Weisen Liu[†], Zeqi Lai[†‡], Qian Wu[†‡], Hewu Li[†‡], Qi Zhang[‡], Zonglun Li[†], Yuanjie Li[†‡], Jun Liu[†‡]
[†]*Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing 100084, China*
[‡]*Zhongguancun Laboratory, Beijing, China*

***Abstract*—With the rapid evolution of space-borne capabilities, *space edge computing (SEC)* is becoming a new computation paradigm for future integrated space and terrestrial networks. Satellite edges adopt advanced on-board hardware, which not only enables new opportunities to perform complex intelligent tasks in orbit, but also involves new challenges due to the additional energy consumption in power-constrained space environment.**

In this paper, we present PHOENIX, an energy-efficient task scheduling framework for emerging SEC networks. PHOENIX exploits a key insight that in the SEC network, there always exist a number of *sunlit edges* which are illuminated during the entire orbital period and have sufficient energy supplement from the sun. PHOENIX accomplishes energy-efficient in-orbit computing by judiciously offloading space tasks to "sunlight-sufficient" edges or to the ground. Specifically, PHOENIX first formulates the *SEC battery energy optimizing (SBEO)* problem which aims at minimizing the average battery energy consumption while satisfying various task completion constraints. Then PHOENIX incorporates a sunlight-aware scheduling mechanism to solve the SBEO problem and schedule SEC tasks efficiently. Finally, we implement a PHOENIX prototype and build an SEC testbed. Extensive data-driven evaluations demonstrate that as compared to other state-of-the-art solutions, PHOENIX can effectively reduce up to 54.8% SEC battery energy consumption and prolong battery lifetime to $2.9\times$ while still completing tasks on time.

I. INTRODUCTION

With the rapid evolution in the aerospace industry, emerging low earth orbit (LEO) satellite mega-constellations not only extend the network boundary of today's terrestrial Internet, but also spawn an innovative computation paradigm: **"space edge computing (SEC)"**. Based on advanced on-board hardware, emerging SEC technologies combine the capabilities of satellite communication and edge computing to provide edge-like services right at the satellite [1]–[3], and further enable a series of intelligent space applications such as smart remote sensing [4], autonomous debris detection and avoidance [5], and Internet of space things [6], *etc.*

While SEC has broad application prospects, it also involves new technical challenges in the energy-constrained outer space environment. On the one hand, fully realizing the promising capability of SEC requires extra advanced on-board hardware to support complex space missions, *e.g.,* exploiting satellite GPUs [7], [8] for data inference and deploying high-speed inter-satellite communication links (ISL) for cross-edge collaborative processing [9]. On the other hand, these additional payloads inevitably involve more energy consumption on satellite edges. Since the energy usage can significantly affect the execution of SEC tasks as well as the battery lifetime (as we will introduce later in §II), *accomplishing energy-efficient space task execution is undoubtedly a crucial problem for futuristic SEC networks.*

Task offloading, which has been well-studied by the conventional mobile computing community over the past decade, is an effective approach for optimizing energy consumption on power-constrained devices [10], [11]. The core idea behind traditional task offloading is to transfer the energy-intensive tasks from the power-constrained devices to a high-performance server with sufficient power supplement (*e.g.,* a cloud), and receive the results after remote execution. Energy can be saved if the network transmission consumes less energy than local task execution. In an SEC scenario, a straightforward method to apply task offloading for energy-efficiency is to transfer space tasks to a nearby ground station which typically has sufficient computation capability and power supplement. However, due to huge amount of SEC data [12], naively offloading all tasks to the ground can easily overwhelm the satellite downlink [8] and involve high latency which is unacceptable for time-sensitive SEC tasks like satellite-based wildfire monitoring and rescue.

To address the limitation of existing offloading approaches, this paper presents PHOENIX, an energy-efficient and deadline-aware task scheduling framework for emerging SEC networks. The design of PHOENIX stands upon a series of important insights obtained from today's LEO satellite constellations. First, sub-systems in a satellite edge are powered directly by solar panels with sufficient power supplement when the satellite is illuminated by sunlight, and are powered by a rechargeable battery when the satellite enters the earth's shadow. Thus, the key to energy optimization is to reduce the energy consumption of the satellite battery. Second, as satellites move, we observe that there dynamically exist a number of orbit planes where satellites in these orbits are exposed to sunlight with near-100% sunlit ratio. Carrying out computation tasks on these "sunlit edges" does not consume battery power. Third, an SEC task is typically associated with a time-to-completion requirement, especially for time-sensitive applications. Taken them together, PHOENIX accomplishes energy-efficiency by dynamically and judiciously offloading space tasks to computational nodes with sufficient power supplement subjecting to various task deadlines. To this end, PHOENIX makes scheduling decisions based on the following options: (i) processing the data locally on a satellite edge; (ii) offloading SEC tasks to other proper "sunlit edges"; or (iii) offloading SEC tasks to ground stations.

Specifically, PHOENIX calculates the decisions in two steps. First, given the SEC network information and the completion

time requirements of various tasks, PHOENIX formulates the *SEC Battery Energy Optimization (SBEO)* problem which targets at minimizing the maximum depth-of-discharge (DoD) of all satellite edge batteries, while satisfying various network, computation and application-level constraints. DoD is an important metric that characterizes the energy usage of a battery and can affect the lifetime of the rechargeable battery as well as the satellite itself. However, efficiently solving the SBEO problem is non-trivial since we prove its NP-hardness and multiple concurrent space tasks can compete for the dynamic computation and network resources in the SEC network.

Second, PHOENIX incorporates a sunlight-aware dynamic SEC task scheduling mechanism which decomposes the original SBEO problem and adopts a series of heuristic algorithms to calculate appropriate scheduling decisions efficiently. Specifically, to reduce the problem complexity, PHOENIX jointly combines coarse-grained orbit-level and find-grained per-satellite task scheduling to calculate near-optimal task assignments.

We build a data-driven hardware-in-the-loop SEC testbed and implement a PHOENIX prototype upon it. Our testbed integrates large-scale SEC network simulation, and low-power computational hardware that has been verified in real space environments. Extensive evaluations demonstrate that PHOENIX can outperform other state-of-the-art SEC approaches in terms of energy consumption, battery lifetime, task deadline satisfaction *etc.*, under various experiment configurations.

Contributions of this paper can be summarized as follows: (i) we formulate the SEC battery energy optimization (SBEO) problem and expose the technical challenges of solving it efficiently and effectively; (ii) we propose PHOENIX, a novel sunlight-aware energy-efficient task scheduling framework for optimizing satellite battery usage and extending the lifetime of SEC networks; (iii) we implement a PHOENIX prototype and conduct extensive data-driven, hardware-in-the-loop experiments to demonstrate the effectiveness of PHOENIX.

II. TECHNICAL BACKGROUND

A. Space Edge: A New Intelligent Computing Paradigm

On-board hardware evolution in aerospace industry. On-board network and computation capabilities have increased significantly over the past decade. The high-speed inter-satellite and ground-satellite links (*i.e.,* ISLs and GSLs) have been successfully deployed on satellite platforms, which can offer Gbps-level data transmission for satellite communication [13], [14] and facilitate the inter-networking of a large number of satellites. Besides, on-board processing capability has also been promoted via low-power commercial off-the-shelf (COTS) hardware accelerators, *e.g.,* graphics processing unit (GPU) [8], vision processing unit (VPU) [15], and field programmable gate array (FPGA) [16]. In 2020, European Space Agency (ESA) launched the Φ-sat-1 sensing satellite equipped with Intel Movidius Myriad 2 to verify the on-board performance of deep neural network (DNN) and achieved great success [17].
Space edge computing (SEC) enables in-orbit intelligence. With such space hardware evolution above, recently we have witnessed a new intelligent computation paradigm: *space edge computing (SEC)*. SEC exploits edge-like computing capabilities on satellites and within satellite networks to process data in orbits. SEC can be used in many innovative space applications such as autonomous remote decisions, in-orbit detection for disaster discovery, climate monitoring and maritime rescue [18]–[20]. SEC enables faster data processing, reduced latency, and improved efficiency by handling data in orbit rather than sending all data back to the ground.

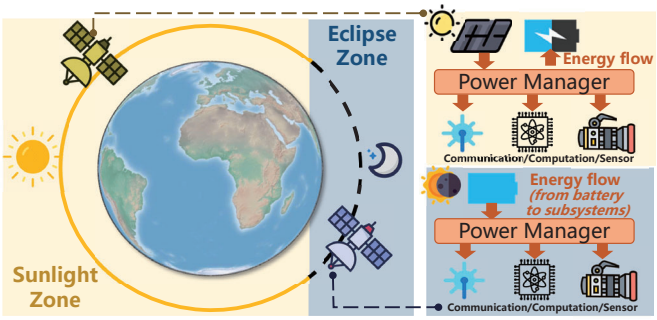
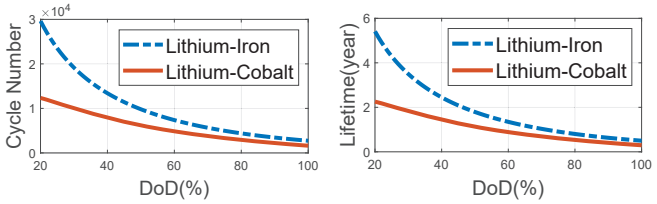


Fig. 1: A typical architecture of SEC on-board power system.

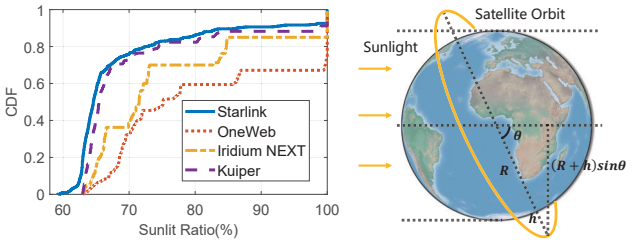


(a) Impact of DoD on cycle number. (b) Impact of DoD on battery lifetime.

Fig. 2: The impact of DoD on the total number of charging-discharging cycles and battery lifetime.

B. Battery Energy Consumption: The Achilles' Heel to SEC

However, while the advanced on-board hardware provides intelligence, it also involves additional energy consumption, which not only affects the execution of SEC tasks, but also affects the lifetime of the SEC system itself.
Satellite power supplement. Satellites orbit around the earth, and during a portion of their orbit, they enter the earth's shadow, causing an eclipse. Fig. 1 plots a typical architecture of existing SEC power systems (SPS). During the sunlight phase, solar panels generate energy from the received photons, and SPS distributes power to other subsystems of the satellite. On-board rechargeable batteries store excess power generated by solar panels when the satellite is in sunlight, and the stored energy is used to power the satellite during eclipse periods. Typically, among all subsystems the communication and computation modules can contribute copious energy consumption [7].
Depth of discharge (DoD) and battery lifetime. The lifetime of a battery (as well as the satellite itself) is tightly affected by its power usage during the charging-discharging cycles of the battery. In particular, DoD characterizes the percentage of discharged energy relative to the maximum capacity. DoD is equal to 0 if the battery is full and 100% if the battery is empty. As the battery is used, the maximum capacity of the battery will gradually decay. Typically, when the maximum battery capacity falls below 80% of its initial volume, the battery should be retired, which is defined as its lifetime [21], [22]. Previous



(a) CDF of sunlit ratio. (b) Example of 100% sunlit ratio.

Fig. 3: Sunlit ratio analysis for representative constellations.

works [23], [24] have uncovered that if we regularly discharge a battery at a higher DoD, its lifetime will be shorter (*e.g.,* about 20% DoD increase can reduce the lifetime by half). Fig. 2 shows how DoD affects the total number of available charging-discharging cycles and the lifetime of two kinds of Lithium batteries. Considering that battery usage can jointly affect task performance and satellite lifespan, optimizing the battery energy consumption and accomplishing energy-efficient in-orbit task processing is important for futuristic SEC networks.

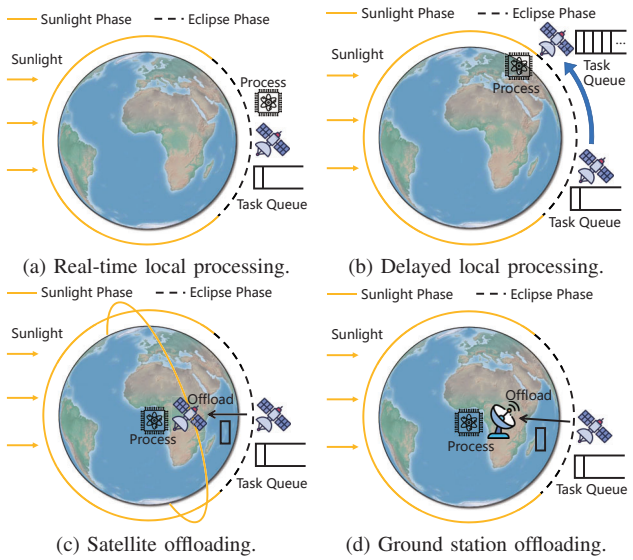
III. PHOENIX DESIGN OVERVIEW

In this section, we introduce the key idea and overview of our sunlight-aware energy-efficient task scheduling strategy.

A. Observation: Sunlight-Sufficient Space Edges

Based on the SPS knowledge introduced in §II-B, we know that the key to optimizing energy usage and prolonging lifetime for SECs is to reduce the *battery energy consumption* caused by in-orbit task processing. To this end, our PHOENIX design leverages an important observation: in an LEO satellite constellation, typically there exist many *"sunlight-sufficient"* satellites which spend most of their orbital time in sunlight.

To quantitatively introduce and understand this phenomenon, we define a metric called *sunlit ratio*, which is calculated by the ratio of the period a satellite is in sunlight to its total operation period. Fig. 3a plots the CDF of sunlit ratio of four state-of-the-art LEO satellite constellations which differ in their orbital altitudes and inclinations. We calculate the sunlit ratio based on their public real-world satellite trajectories [25] during May 2023. We obtain three important observations. First, we find that the sunlit ratio is approximately higher than 60%, indicating that a large fraction of satellites are exposed to the sun for a long time on their orbits. Second, interestingly we observe that some satellites in certain orbits can achieve near-100% sunlit ratio with sufficient sunlit supplement. Fig.3b plots an example to explain this observation. Suppose that radius of the earth is R. The height of satellite orbit is denoted as h and the angle between sunlight and orbital plane is denoted as θ. Then, the distance between the orbit and the earth's core is $R+h$ and the vertical component of distance perpendicular to sunlight is $(R+h)\cdot sin\theta$. If the length of the vertical component is longer than the radius of the earth, the whole orbit can be exposed to the sunlight and the sunlit ratio can reach 100%.



(a) Real-time local processing. (b) Delayed local processing.

(c) Satellite offloading. (d) Ground station offloading.

Fig. 4: The trade-space of SEC task scheduling in PHOENIX.

B. Basic Idea: Sunlight-Aware SEC Task Scheduling

Inspired by the above crucial observations, PHOENIX exploits a key idea to accomplish energy-efficient SEC: *dynamically offloading in-orbit tasks to appropriate power-sufficient nodes* (e.g., *sunlight-sufficient satellites with near-100% sunlit ratio) without exceeding various task deadlines.* Specifically, PHOENIX judiciously schedules SEC tasks based on the following options (also illustrated in Fig. 4) to minimize the battery energy consumption while meeting various requirements of task completion time:

- **Real-time local processing** (Fig. 4a). Once in-orbit data is acquired on a satellite, processing it locally and immediately no matter where the current satellite is.
- **Delayed local processing** (Fig. 4b). Once in-orbit data is acquired, the satellite delays the data processing until a specific time point (*e.g.,* when the satellite leaves eclipse).
- **Offloading tasks to nearby sunlit edges** (Fig. 4c). Instead of being processed locally, space data collected in orbit is transferred to another sunlight-sufficient satellite in the SEC network for energy-efficient processing.
- **Offloading tasks to available ground stations** (Fig. 4d). Space data is transferred to a ground station with sufficient power and computation capability through the SEC network.

Essentially, the above scheduling options represent different preferences on saving battery energy consumption and guaranteeing task completion time. For example, immediate local data processing may achieve shorter mission completion time, but possibly at the cost of higher battery energy consumption. Transferring the entire in-orbit task to the ground may save energy for a satellite edge, but offloading high-volume space data can easily overwhelm the limited downlink and involve unacceptable latency. PHOENIX dynamically schedules various SEC tasks under different computation, network and sunlight conditions, and further makes proper scheduling decisions.

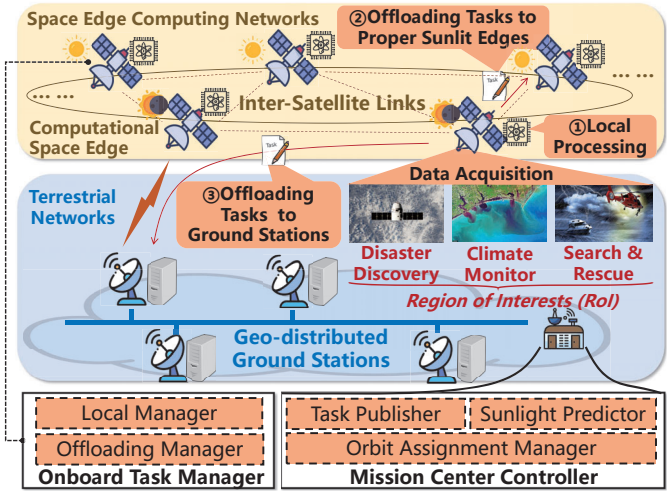


Fig. 5: PHOENIX system overview.

C. System Overview

Architecture. Fig. 5 shows the system overview of PHOENIX, which combines: (i) a swarm of computational space edges constructing an SEC network, and (ii) terrestrial infrastructures such as geo-distributed ground stations and a mission control center. Each satellite is equipped with high-speed ISLs and GSLs for inter-satellite and ground-satellite communication. In addition, an on-board intelligent processor is deployed on each satellite for in-orbit computing. Based on this baseline SEC architecture, PHOENIX accomplishes energy-efficient SEC task scheduling by incorporating two new components as follows.

- **Centralized mission center controller.** The controller is a centralized coordinator, which consists of three modules: task publisher, sunlight predictor and orbit assignment manager. The task publisher distributes the tasks to satellites. The sunlight predictor predicts the sunlight states of satellites based on their trajectories and distributes the sunlight information to satellites for offloading decisions. The orbit assignment manager pre-allocates alternative orbit sets for satellites to avoid resource competition between orbits.
- **Distributed on-board task manager.** The manager on each satellite consists of two modules: local processing manager and offloading manager. The offloading manager decides where to process the task based on the pre-allocated orbit set. The local processing manager arranges the task execution time if a task is decided to be processed locally.

Workflow. SEC tasks are scheduled by the following steps:

- **Task publication and orbit assignment.** The mission center receives tasks from customers, such as requirements of using SEC for disaster discovery, climate monitoring, maritime search and rescue. The common feature of these tasks is that they all require a constellation of satellites persistently capture information (*e.g.,* high-resolution images) of a certain region of interest (RoI). Then, the mission center predicts the sunlight states of satellites and publishes the RoI along with sunlight information to all satellites. Based on the sunlight and task information, the mission center pre-assigns an orbit subset for each orbit and distributes the assignment decision.
- **Task offloading.** When a satellite flys over RoI, it captures images and selects offloading destination for each task, either a ground station or a satellite (including itself, *i.e.,* local processing). The satellite first checks whether the task can be offloaded to a ground station before deadline. If not, it tries to arrange the task locally and judges whether it can finish the task under sunlight phase as well as before deadline. If both of the above conditions can not be satisfied, it offloads the task to other satellite.
- **Processing time arrangement.** The final destination satellite arranges the task processing time (the ground station can process the task immediately after offloading) and sends the result back to the mission center when the task is finished.

IV. ENERGY-EFFICIENT TASK SCHEDULING BY PHOENIX

In this section, we introduce PHOENIX's sunlight-aware energy-efficient task scheduling mechanism in detail.

A. Formulating The SEC Battery Energy Optimization Problem

SEC network topology. Assume that time is slotted with each timeslot ΔT and the time set is denoted as $\mathcal{T} = \{1, 2, \ldots, T_{max}\}$. Denote $G_t = (V, E_t)$ as the satellite network topology. V is the node set and E_t is the edge set at timeslot t. There are two types of nodes, *i.e.,* satellites and ground stations. Assume that the total number of orbits is M and the satellite set in orbit i can be denoted as ORB_i. Thus, the satellite of the whole constellation can be represented by $SAT = \cup_{1 \le i \le M} ORB_i$. The number of ground stations is N and the ground station set is denoted as $GS = \{g_1, g_2, \ldots, g_N\}$. Then the node set can be described by $V = SAT \cup GS$. As satellites orbit around the earth, the visibility between nodes changes over time. Let binary variable $Vis^t_{i,j}$ denote the visibility between node i and node j. If i and j are visible at time t, then $Vis^t_{i,j} = 1$ and they can establish a link (i, j). Therefore, we can check whether a satellite s can connect to any ground station at time t by a binary value $l_{s,t}$:

$$l_{s,t} = 1 - \prod_{g \in GS} (1 - Vis^t_{s,g}). \tag{1}$$

Denote λ as the number of ISLs in each satellite. Typically, there are four ISLs and one GSL for each satellite [26]. The link capacity of (i, j) is denoted as $Cap(i, j)$ and we set $Cap(i, i) = \infty$, meaning that if a task is processed locally, the transmission can be finished in situ instantly.

Task scheduling. Assume that the task set is $TASK$ and a task k can be represented by a tuple $(src_k, z_k, T^k_{arv}, T^k_{cp}, T^k_{ddl})$, indicating the source satellite creating the task, the task size, the task arrival time, the computing time required to process the task for satellite, and the deadline respectively. A task can be offloaded to ground stations, processed locally or offloaded to another satellite. Let dst_k denote the node where task k is processed. If $dst_k \in GS$, task k is downloaded by ground stations directly; if $dst_k = src_k$, task k is processed locally; otherwise, task k is offloaded to another satellite. Then, the tasks that select node s as the processing destination can be represented by $\mathcal{K}_s = \{k | dst_k = s, k \in TASK\}$. Let T^k_{of} be

the point of time when task k finishes offloading. The task offloading time is decided by the bottleneck of path from src_k to dst_k. Given a routing path $R(i,j,t)$ from node i to node j at timeslot t, the offloading path can be denoted as $R(src_k, dst_k, t)$. We use a binary variable $r_{i,j}^{k,t}$ to indicate whether the offloading flow of k goes through link (i,j) at t:

$$r_{i,j}^{k,t} = \begin{cases} 1, (i,j) \in R(src_k, dst_k, t) \\ 0, otherwise \end{cases}. \quad (2)$$

Assume that all flows on the same link share the link capacity fairly. Then, the bandwidth that each flow can obtain on link (i,j) at timeslot t is $Cap(i,j)/\sum_{k \in TASK} r_{i,j}^{k,t}$. Denote $sz_{k,t}$ as the data size of task k transmitted at timeslot t. The data size that can be transmitted in each timeslot is limited by the minimum bandwidth on the offloading path:

$$sz_{k,t} = \Delta T \cdot \min\{\frac{Cap(i,j)}{\sum_{\sigma \in TASK} r_{i,j}^{\sigma,t}} | r_{i,j}^{k,t} = 1, i, j \in V\}. \quad (3)$$

When the amount of sent data reaches the task size, the offloading is finished and the time to finish offloading is:

$$T_{of}^{k} = \min\{\tau | \sum_{t=T_{arv}^{k}}^{\tau} sz_{k,t} \geq z_k, \tau \in \mathcal{T}\}. \quad (4)$$

After the task is offloaded, the receiver should decide when to process the task. Let T_{bcp}^{k} denote the time to begin processing and the time to complete processing can be represented by $T_{bcp}^{k} + T_{cp}^{k} - 1$. We use a binary variable $x_{k,t}$ to indicate whether the task k is being processed at time t:

$$x_{k,t} = \begin{cases} 1, & T_{bcp}^{k} \leq t \leq T_{bcp}^{k} + T_{cp}^{k} - 1 \\ 0, & otherwise \end{cases}. \quad (5)$$

Thus, we denote the task processing matrix as $\mathcal{X} = \{x_{k,t} | k \in TASK, t \in \mathcal{T}\}$. For any satellite s, the number of tasks under processing at timeslot t can be calculated by $\sum_{k \in \mathcal{K}_s} x_{k,t}$.

Energy Consumption. As mentioned in § II-B, the electronic components in satellites consist of solar panels, communication terminals, processing units, battery and other basic modules (*e.g.,* sensors). For satellite s, we denote the power generated by its solar panels under sunlight phase as P_{solar}^{s}. As satellites enter eclipse phase and sunlight phase alternately, we use binary variable $sun_{s,t}$ to indicate whether satellite s is illuminated at time t. Thus, the power generated by solar panels at timeslot t is $sun_{s,t} \cdot P_{solar}^{s}$. We use P_{ISL}^{s} and P_{GSL}^{s} to describe the transmit power of ISL and GSL respectively. Therefore, the power consumed by ISL and GSL at timeslot t are $\lambda \cdot P_{ISL}^{s}$ and $l_{s,t} \cdot P_{GSL}^{s}$ respectively. Denote the power of processing units as P_{cp}^{s}. Thus, the power consumed by processing units at timeslot t is $P_{cp}^{s} \cdot \sum_{k \in \mathcal{K}} x_{k,t}$. The power consumed by other basic modules is denoted as P_{basic}^{s}. Let B_{vol}^{s} be the volume of battery and $B_{s,t}$ be the rest battery energy of s at time t, which is decided by $\mathcal{K}_s$ and $\mathcal{X}$. In the beginning, the battery is full, *i.e.,* $B_{s,0} = B_{vol}^{s}$. As time goes by, the rest battery energy changes according to the energy produced by solar panels and the energy consumed by electronic devices:

$$\begin{aligned} B_{s,t} = \min\{(sun_{s,t} \cdot P_{solar}^{s} - P_{basic}^{s} - P_{cp}^{s} \cdot \sum_{k \in \mathcal{K}_s} x_{k,t} \\ - \lambda \cdot P_{ISL}^{s} - l_{s,t} \cdot P_{GSL}^{s}) \cdot \Delta T + B_{s,t-1}, B_{vol}^{s}\}. \end{aligned} \quad (6)$$

Therefore, the DoD of satellite s at timeslot t can be represented by $1 - B_{s,t}/B_{vol}^{s}$.

SEC Battery Energy Optimization (SBEO) problem formulation. Given the following inputs: (i) time set $\mathcal{T}$ and timeslot duration ΔT; (ii) node set V and visibility $Vis_{i,j}^{t}$ between nodes; (iii) link capacity $Cap(i,j)$; (iv) task set $TASK$ and routes between nodes $R(i,j,t)$; (v) power of electronic devices P_{solar}^{s}, P_{basic}^{s}, P_{ISL}^{s}, P_{GSL}^{s} and P_{cp}^{s}; (vi) battery volume B_{vol}^{s} and sunlight indicator $sun_{s,t}$, we aim to provide the processing task set $\mathcal{K}_s$ for all satellites and the processing matrix $\mathcal{X}$ such that the maximum DoD among all satellites is minimized, prolonging the lifetime of satellites as illustrated in § II-B:

$$\min \max_{s \in SAT, t \in \mathcal{T}} 1 - B_{s,t}/B_{vol}^{s} \quad (7)$$

Constraints:

(i) A satellite can only process a task at each timeslot:

$$\sum_{k \in \mathcal{K}_s} x_{k,t} \leq 1, \forall s \in SAT, t \in \mathcal{T}. \quad (8)$$

(ii) Task processing should be scheduled after offloading finish:

$$T_{of}^{k} \leq T_{bcp}^{k}, \forall k \in TASK. \quad (9)$$

(iii) Task processing should be completed before deadline:

$$T_{bcp}^{k} + T_{cp}^{k} \leq T_{ddl}^{k}, \forall k \in TASK. \quad (10)$$

Complexity analysis. To analyze the complexity, we simplify the SBEO problem to an easier case where ground stations are not considered and there is no transmission cost. Then, a task k can be divided into T_{cp}^{k} slots and a satellite s can be divided into T_{max} slots. There are two types of satellite slots according to the sunlight state, *i.e.,* sunlight slots and eclipse slots. If a task slot is assigned to a eclipse slot of a satellite, the cost is 1 while there is no cost when assigned to a sunlight slot. Then, the problem can be transformed into a generalized assignment problem, which assigns the task slots to satellite slots, aiming to minimize the cost. The generalized assignment problem has been proven to be NP-hard [27]. Thus, the simplistic problem as well as the SBEO problem are NP-hard.

B. Sunlight-Aware Dynamic SEC Task Scheduling Algorithms

To solve the SBEO problem, we decompose the problem into three parts based on the PHOENIX architecture proposed in §III-C: predetermined orbit assignment, on-board offloading selection and processing arrangement. First, we propose a sunlight-aware orbit assignment algorithm (Algorithm 1) running in the mission center before task arrival. The mission center calculates an alternative orbit subset (denoted as alt_set) for each orbit as the input of the offloading selection algorithm. Satellites can only offload tasks to ground stations or orbits in

Algorithm 1: Sunlight-aware Orbit Assignment

Input: topology G_t, task set $TASK$, indicator $sun_{s,t}$
Output: alternative subset alt_set

1 $cyc \leftarrow$ `GetOrbitalCycle()`
2 $\mathcal{A}_s \leftarrow \{k | src_k = s, t \le T^k_{arv} \le t+cyc-1\}, \forall s \in SAT$
3 */* estimate sunlight duration and task amount. */*
4 **for** *each orbit* $i \leftarrow 1, 2, \ldots, M$ **do**
5 $\quad sunlit[i] \leftarrow \sum_{s \in ORB_i} \sum_{\tau=t}^{t+cyc-1} sun_{s,\tau}$
6 $\quad task[i] \leftarrow \sum_{s \in ORB_i} \sum_{k \in \mathcal{A}_s} T^k_{cp}$
7 **for** *each orbit* $i \leftarrow 1, 2, \ldots, M$ **do**
8 $\quad w[i] \leftarrow task[i] / \sum_{j=1}^{M} task[j]$
9 $\quad target[i] \leftarrow$ `Int`$(w[i] * \sum_{j=1}^{M} sunlit[j]) - sunlit[i]$
10 */* assign orbit subset. */*
11 $idle \leftarrow \{i | task[i] = 0\}, \forall i, 1 \le i \le M$
12 **for** *each orbit* $i \leftarrow 1, 2, \ldots, M$ **do**
13 $\quad$ **if** $target[i] < 0$ **then**
14 $\quad\quad alt\_set[i] \leftarrow \{i\}$
15 $\quad$ **else**
16 $\quad\quad subset \leftarrow$ `Knapsack`$(idle, sunlit, target[i])$
17 $\quad\quad alt\_set[i] \leftarrow \{i\} \cup subset$
18 $\quad\quad idle \leftarrow idle - subset$
19 **return** alt_set

Algorithm 2: Orbit-based Offloading

Input: satellite src, task k, alternative subset alt_set
Output: offloading node dst_k, processing matrix $\mathcal{X}$

1 */* offload to ground station. */*
2 $gs, T_{gs} \leftarrow$ `GetAvailableGS`(t)
3 **if** $T_{gs} + z_k / Cap(src, gs) \le T^k_{ddl}$ **then**
4 $\quad T_{gs} \leftarrow T_{gs} + z_k / Cap(src, gs)$
5 $\quad dst_k \leftarrow gs$, **continue**
6 $\hat{\mathcal{X}}, flag\_sun \leftarrow$ `Arrange`$(src, t, \mathcal{K}_{src} \cup \{k\})$
7 **if** $flag\_sun == 1$ **then**
8 $\quad dst_k \leftarrow src$ */* process locally. */*
9 **else**
10 $\quad$ */* offload to other satellite. */*
11 $\quad cnt[] \leftarrow$ `GetTaskCounter()`
12 $\quad i \leftarrow \arg\min_{j \in alt\_set[src.orbit]} cnt[j] / sunlit[j]$
13 $\quad cnt[i] \leftarrow cnt[i] + T^k_{cp}$
14 $\quad E[s] \leftarrow$ `QueryEnergy`$(s), \forall s \in ORB_i$
15 $\quad dst_k \leftarrow \arg\max_{s \in ORB_i} E[s]$
16 **if** $dst_k == src$ **then**
17 $\quad \mathcal{X} \leftarrow \hat{\mathcal{X}}$ */* confirm local processing time.*/*
18 **return** dst_k, $\mathcal{X}$

their alternative subset, which can avoid competition among orbits. Second, we propose an orbit-based offloading algorithm (Algorithm 2) running in each satellite to decide the processing matrix $\mathcal{X}$ and offloading destination dst_k, which invokes the processing arrangement algorithm (Algorithm 3). Note that the output of SBEO problem $\mathcal{K}_s$ can be constructed from dst_k. Last, the processing arrangement algorithm decides when to process tasks, minimizing the battery energy consumption while avoiding deadline miss. All of the three algorithms can be solved in polynomial time, thus we can solve the SBEO problem in polynomial time via such decomposition.

Orbit assignment in mission center. The key ideas are summarized as follows: (i) To fairly assign the orbit subsets, the energy capability of each subset should match the task amount. Therefore, we use sunlight duration in a period to represent the energy capability and select the orbit subsets which satisfy the following properties: the subsets for orbits generating tasks are disjoint and the energy capability proportion of each subset is close to the task proportion. (ii) To reduce the complexity of searching suitable subsets, we set up a target capability for each subset according to task amount and transfer the problem to a knapsack problem, which finds an orbit subset that minimizes the gap to target capability.

Algorithm 1 shows the details of sunlight-aware orbit assignment algorithm. First, function `GetOrbitalCycle` calculates the orbital period cyc, the time for a satellite to complete one orbit (line 1). $\mathcal{A}_s$ denotes the tasks generated by satellite s in next period. Then, we predict the sunlight duration and task amount of each orbit in the next period, represented by $sunlit[i]$ and $task[i]$ for orbit i respectively (line 4-6). $w[i]$ records the task amount proportion of orbit i (line 8) and we expect to find a subset with similar energy capability proportion. So we set up the target capability (denoted as $target[i]$) for orbit i based on the task amount proportion. The alternative subset for each orbit must include itself, thus we subtract its energy capability and convert the target capability into an integer for knapsack problem (line 9). We use $idle$ to represent the set of orbits without task generation. If the target capability is less than 0, the orbit can self-satisfy the energy requirement, so the alternative set is itself (line 14). Otherwise, we regard orbits as items associated with weights $sunlit$ and call function `Knapsack` to select orbits from $idle$ such that the gap to bag capacity $target[i]$ is minimized (line 16). In the knapsack problem, the number of items is at most M and bag capacity $target[i]$ is at most $cyc \cdot |SAT|$. Therefore, the time complexity of Algorithm 1 is $\mathcal{O}(M^2 \cdot cyc \cdot |SAT|)$.

Offloading selection in each satellite. Based on the alternative subset assigned by Algorithm 1, the orbit-based offloading algorithm adopts the following ideas: (i) As the connection of GSLs can be predicted, we first check whether tasks can be offloaded to ground stations before deadline to save on-board computation resources. (ii) For satellite offloading, we maintain a task counter for orbits in the alternative subset and try to keep the task amount conforming to their energy capability.

Algorithm 2 shows the details of the orbit-based offloading algorithm. When a new task k arrives at satellite src, the offloading manager obtains the nearest available ground station gs and its available time T_{gs} via function `GetAvailableGS` (line 2). If the task can be offloaded to gs before deadline, we select gs as the offloading destination and update the available time (line 3-5). Note that we do not adopt multi-hop ground station offloading due to the re-routing problem

caused by GSLs change. If not, we try to arrange the task locally via Algorithm 3, searching for the possible processing matrix $\hat{\mathcal{X}}$ and checking whether the task can be processed completely in sunlight (denoted as $flag\_sun$). If $flag\_sun$ is 1, we process the task locally (line 8). Otherwise, we offload the task to other satellite. We obtain the task counter array cnt via function `GetTaskCounter`, which is initialized to 0 in the first timeslot. Then, we select the orbit with minimum ratio of task counter to sunlight duration and update its task counter (line 12-13). The source satellite sends queries to satellites in the selected orbit and each satellite receiving the query responds with its energy state $E[s]$, which is decided by sunlight duration, rest battery energy and task queue:

$$E[s] = P^s_{solar} \sum_{\tau=t}^{t+cyc-1} sun_{s,t} + B_{s,t-1} - P^s_{cp} \sum_{k \in \mathcal{K}_s} T^k_{cp}. \quad (11)$$

Then, the source satellite chooses the one with maximum energy as the offloading node (line 15). Finally, if the task is processed locally, we adopt the processing arrangement calculated before to avoid recomputation (line 17).

Processing arrangement in each satellite. To arrange the processing time, we apply the deadline first scheme and search for the delay to exploit as less eclipse timeslots. The details of processing arrangement algorithm are shown in Algorithm 3. First, we sort the tasks according to their deadlines in ascending order. Then, we calculate the latest time of task k to start processing (denoted as T^k_{latest}), which guarantees the deadline requirement (line 2-3). Next, we calculate the earliest time of task k to start processing (denoted as $T^k_{earliest}$). The indicator $flag\_sun$ is initialized to 1 and the earliest time of the first task is initialized to current time t. Then, we search for the next sunlit time T^k_{sunlit} (line 5). If the next sunlit time is before the latest task processing time, we arrange the task to be processed from T^k_{sunlit} to save battery energy (line 6). Otherwise, we arrange the task to be processed from T^k_{latest} to guarantee the deadline requirement (line 8) and set $flag\_sun$ to 0 (line 8). At the end of each loop, we update the earliest processing time of next task (line 9).

Algorithm 3: Processing Arrangement

1 **Function** `Arrange`$(s,t,\mathcal{K}_s)$**:**
2 `Sort`$(\mathcal{K}_s)$ based on T^k_{ddl} in ascending order
$T^{|\mathcal{K}_s|}_{latest} \leftarrow T^{|\mathcal{K}_s|}_{ddl} - T^{|\mathcal{K}_s|}_{cp}$ **for**
$k \leftarrow |\mathcal{K}_s|-1, |\mathcal{K}_s|-2, \ldots, 1$ **do**
3 $\quad T^k_{latest} \leftarrow \min(T^k_{ddl}, T^{k+1}_{latest}) - T^k_{cp}$
4 $flag\_sun \leftarrow 1,\ T^1_{earliest} \leftarrow t$ **for**
$k \leftarrow 1, 2, \ldots, |\mathcal{K}_s|$ **do**
5 $\quad T^k_{sunlit} \leftarrow \min\{\tau | sun_{s,\tau} = 1, \tau \geq T^k_{earliest}\}$ **if**
$T^k_{sunlit} \leq T^k_{latest}$ **then**
6 $\quad\quad x_{k,\tau} \leftarrow 1, \forall \tau, T^k_{sunlit} \leq \tau < T^k_{sunlit} + T^k_{cp}$
7 $\quad$ **else**
8 $\quad\quad x_{k,\tau} \leftarrow 1, \forall \tau, T^k_{latest} \leq \tau < T^k_{latest} + T^k_{cp}$
$\quad\quad flag\_sun \leftarrow 0$
9 $\quad T^{k+1}_{earliest} \leftarrow \min(T^k_{sunlit}, T^k_{latest}) + T^k_{cp}$
10 **return** $\mathcal{X}, flag\_sun$

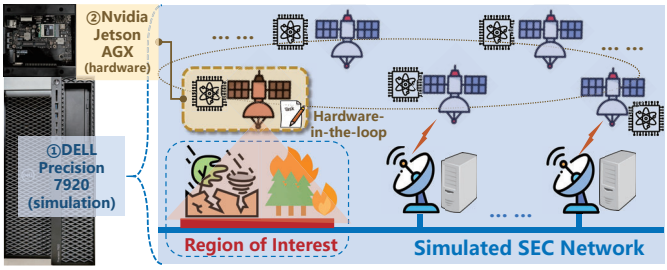


Fig. 6: Our hardware-in-the-loop testbed combines: (1) a Nvidia Jetson low-power edge computing hardware to mimic a real satellite edge, and (2) a high-end server simulating a large-scale SEC network, distributed ground stations *etc.*

V. Performance Evaluation

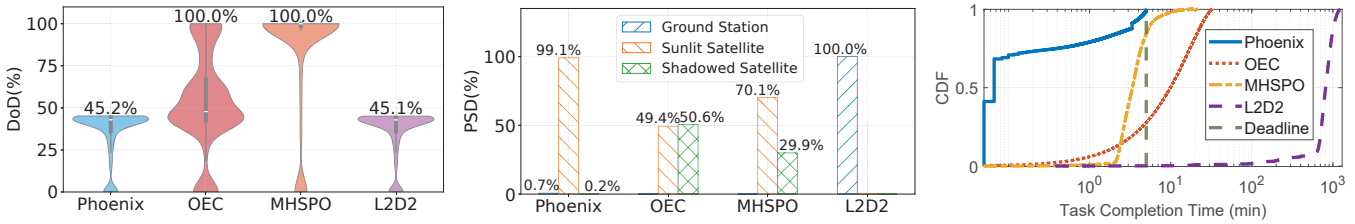
In this section, we implement an SEC testbed and evaluate the performance of PHOENIX to illustrate its effectiveness. First, we compare the DoD of satellite batteries and task completion time with the other state-of-the-art strategies. Second, we explore the performance under four seasons as the revolution of the earth can affect the sunlit ratio defined in §III-A. Third, we apply the strategies to different constellations to explore the impact of constellation parameters. Finally, we setup various workloads to illustrate the robustness of PHOENIX via tuning the processing capability and task type.

A. Environment Setup

Prototype implementation. We build a data-driven hardware-in-the-loop SEC testbed based on StarryNet [28], a recent container-based satellite network emulator. The SEC environment is deployed on a Dell Precision 7920 Tower Workstation connected with a Jetson AGX Orin Developer Kit as shown in Fig. 6. The Developer Kit works as an SEC node, running machine learning models to evaluate the task completion time and energy consumption. Based on the +Grid [26] structure and the trajectory of the satellite simulated by the Developer Kit, it establishes virtual links with its adjacent satellites and ground facilities dynamically.

LEO constellation settings. We conduct extensive simulation driven by real-world information, including two kinds of LEO constellations: inclined orbit constellation (Starlink [29]) and polar constellation (OneWeb [30]). Following [12], we use the ground station locations collected by SatNOGS [31]. For computation, we adjust the power level of Jetson AGX Orin Developer Kit to 30W/50W/60W, providing different computation capabilities. Based on the existing hardwares [32], [33], we set the power of GSL/ISL to 16W/10W respectively. Following [7], we set the basic power to 4W. And we set the power generated by solar panels to 120W and the battery volume to 60Wh, which can offer sufficient energy. For transmission, we set the capacity of GSL/ISL to 100Mbps/1Gbps respectively.

SEC tasks and datasets. We select ship detection [1], [19] and wildfire segmentation [34] as the SEC tasks. For ship detection, we apply YOLO [35] to dataset [36] and select Atlantic Ocean



(a) DoD distribution of satellites. (b) The percentage of scheduling decision. (c) CDF of task completion time with log scale.

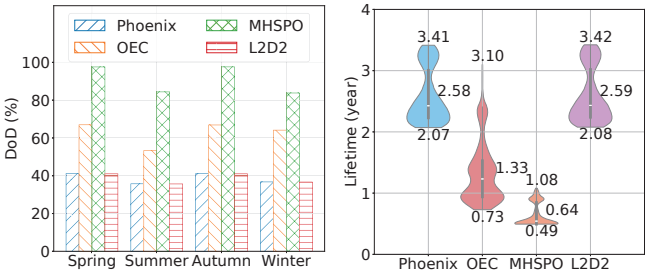
Fig. 7: Performance comparison of different offloading strategies.

as the RoI. Satellites continuously capture images every second, with the resolution of $10K \times 10K$ pixels [8]. Each image is expected to be processed within 5 minutes [37]. Based on our measurement, the processing time of an image for ship detection is 10s/5s/3s under 30W/50W/60W respectively. For wildfire segmentation, we apply U-Net [38] to dataset [39] and select Amazon Rainforest as the RoI. The imaging interval is set to 5 seconds and the processing time of an image is 120s/67s/51s under 30W/50W/60W respectively. Other parameters are the same as ship detection task.

Comparison objects and metrics. We implement three state-of-the-art offloading schemes for comparison: (i) OEC [7], which processes the tasks with an intra-orbit pipeline; (ii) MHSPO [9], an energy-efficient satellite peer offloading scheme, and (iii) L2D2 [12], which offloads tasks to geo-distributed ground stations. We regard L2D2 as the baseline because it doesn't consume any *computation energy*, and thus has the lowest energy consumption. We use DoD, battery lifetime and task finish time as metrics to present the effectiveness of PHOENIX.

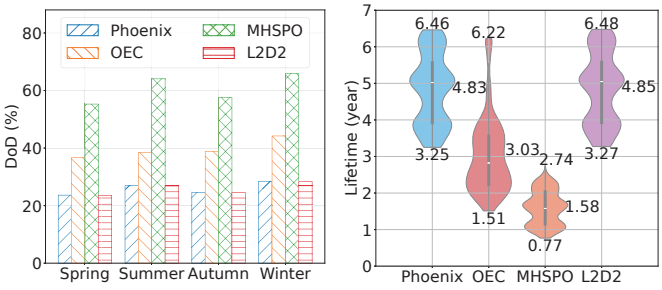
B. DoD and Task Completion Time Comparison

We first compare the performance under the configuration of Starlink constellation, using 60W power level for ship detection. As shown in Fig. 7a, PHOENIX is close to L2D2 and reduces the maximum DoD by 54.8% as compared with OEC and MHSPO. Note that there are some satellites with 0% DoD for all strategies because these satellites can keep illuminated by sun without consuming the battery energy. As tasks can be processed in ground stations, sunlit satellites or shadowed satellites, Fig. 7b plots the percentage of scheduling decisions (PSD), which indicates the ratio of three types of nodes selected to process tasks. We can see that PHOENIX outperforms other on-board computation strategies from two aspects: (i) PHOENIX can cooperatively exploit the communication capability of ground stations and on-board computation capability of satellites; (ii) PHOENIX can exploit the sunlit satellites to process tasks (99.1% tasks are processed in sunlit satellites) while reducing the consumption of shadowed satellites to save the energy of battery (only 0.2% tasks are processed in shadowed satellites). As shown in Fig. 7c, PHOENIX can satisfy the deadline requirement while OEC, MHSPO and L2D2 may miss deadline. As L2D2 offloads tasks to ground stations directly without on-board computing, it consumes the least battery energy, but the task completion time is very high due to the downlink bottleneck. Overall, PHOENIX can not only reduce the DoD of batteries, achieving the near-optimal performance, but also satisfy the deadline requirement.



(a) Average DoD under four seasons. (b) Estimated battery lifetime.

Fig. 8: Performance under Starlink configuration in 2023.



(a) Average DoD under four seasons. (b) Estimated battery lifetime.

Fig. 9: Performance under OneWeb configuration in 2023.

C. Impact of Seasons and Battery Lifetime Extension

As the earth revolves around the sun, the sunlit ratio of each satellite may change over a year. Thus, we repeat the above experiments under the configuration of four seasons, comparing the average DoD in Fig. 8a and estimated lifetime in Fig. 8b. As shown in Fig. 8a, PHOENIX is close to L2D2 and outperforms OEC/MHSPO in any season. Meanwhile, we find that the DoD of all strategies in summer and winter is lower than that in spring and autumn. This is because the angles between some inclined orbits and sunlight are nearly vertical in summer and winter, which can offer larger sunlit ratio. This phenomenon also indicates that the sunlight is an important factor in energy optimization. Following the life model in [23], we estimate the battery lifetime of each satellite when applying different strategies. As shown in Fig. 8b, PHOENIX is close to L2D2 and can prolong the satellite lifespan up to $2.9\times$ /$5.3\times$ as compared with OEC/MHSPO. This is because PHOENIX is sunlight-aware and exploits the sunlit satellites to process tasks, which can significantly reduce the battery energy consumption.

TABLE I: Average DoD under different processing capabilities and two types of workloads.

DoD(%) Strategy / Power		PHOENIX	OEC	MHSPO	L2D2
Ship Detection	30W	43.1	47.5	59.3	35.7
	50W	36.0	53.2	80.5	35.7
	60W	35.7	53.3	84.4	35.7
Wildfire Segmentation	30W	42.1	42.7	58.2	35.7
	50W	41.3	47.0	79.0	35.7
	60W	38.4	48.7	83.9	35.7

D. Impact of Constellation Parameters

The sunlit ratio is also correlated to the parameters of constellations (*e.g.,* inclination, altitude) as mentioned in §III-A. We simulate OneWeb constellation in our testbed and compare the performance of four strategies. As shown in Fig. 9a, PHOENIX is close to L2D2 and outperforms OEC/MHSPO by up to 15.7%/37.4% on average DoD. Different from inclined orbit, the DoD in summer and winter is larger than that in spring and autumn. This is because polar orbit gets more sunlit ratio in spring and autumn, which is opposite to the inclined orbit. Fig. 9b plots the lifetime of satellites, which shows that PHOENIX can achieve longer lifetime to $2.3\times$ /$4.4\times$ as compared with OEC/MHSPO. When applying PHOENIX under different constellation configurations, OneWeb can achieve better performance than Starlink with lower DoD (12.8% on average) and longer lifetime ($1.9\times$ on average) for two key reasons: (i) polar orbit can experience longer sunlight duration than inclined orbit; (ii) the altitude of OneWeb is higher than that of Starlink, which can obtain higher sunlit ratio.

E. Impact of Various Capabilities and Workloads

Finally, we adjust the power of Developer Kit from 30W to 60W and apply wildfire segmentation task to explore the performance under various computation capabilities and workloads. TABLE I shows the average DoD of each strategy under different power levels and workloads. Results show that PHOENIX is close to L2D2 and outperforms others under various computation capabilities and workloads, which indicates the robustness of PHOENIX. The DoD of L2D2 remains unchanged due to no on-board processing. When the power increases, the DoD becomes smaller for PHOENIX, which is somewhat counterintuitive. This is because the processing time of tasks gets shorter with strengthened computation capability. The energy is relative to both power and time, thus the energy consumed by a ship detection (wildfire segmentation) task is 300J/250J/180J (3600J/3350J/3060J) under 30W/50W/60W respectively. This inspires us that even if we promote the computation capability, the battery energy consumption can be reduced via proper task scheduling.

VI. RELATED WORK

Efficient network delivery for big in-orbit data. Emerging satellites with evolved remote sensing capabilities are widely used in many applications such as the earth surveillance and disaster monitoring. A number of recent efforts have studied the approaches for accelerating space data delivery and optimizing the task completion time [12], [40], [41]. L2D2 [12] is a space data download scheme which uses commodity hardware to offer low latency and robust download. OrbitCast [40] is a hybrid space data delivery architecture that collaboratively leverages LEO satellites and geo-distributed ground stations to fast forward space data. However, with the increasing resolution of emerging on-board sensors, the amount of space data has also increased exponentially in recent years. Downloading all space data to the ground requires massive amounts of bandwidth and storage in satellite, which induces large downloading latency.
SEC for in-orbit data processing. Other efforts investigated the feasibility of leveraging edge-like computation capabilities on emerging satellites to directly process data in orbit [7], [8]. This technique, known as space/orbit edge computing, can identify and discard unnecessary information among the big space data. Network efficiency is improved since only high-value data will be downloaded to the ground. However, these works ignore the energy challenge caused by the additional workload of in-orbit processing. More recently, MHSPO [9] leverages Lyapunov optimization to optimize energy consumption for SEC networks by offloading tasks to peer satellites. The fundamental difference between MHSPO and PHOENIX is that MHSPO ignores the time-varying sunlight states of LEO satellites, which weakens the effectiveness of battery energy optimization for SEC networks.
Task scheduling in mobile edge computing. Energy-efficient task scheduling [10], [11], [42], [43] has been well studied in terrestrial networks, which exploit dynamic CPU frequency scaling, network interface selection and transmission power allocation techniques to adaptively make offloading decisions and control energy consumption. However, the large amount of tasks makes them difficult to scale the satellite computation and communication capability. Recent learning-based works [44], [45] have applied deep reinforcement learning in mobile devices to select offloading actions. However, the action space explodes among large-scale satellite agents and selecting actions via deep neural network consumes extra energy.

VII. CONCLUSION

In recent years, *space edge computing (SEC)* is becoming a new computation paradigm for future integrated space and terrestrial networks. In this paper, we present PHOENIX, an energy-efficient task scheduling framework for futuristic SEC networks. PHOENIX exploits a number of *sunlit edges* for on-board task processing. In particular, we propose a series of sunlight-aware scheduling algorithms to reduce battery energy consumption. We implement a PHOENIX prototype and conduct experiments on the SEC environment. Evaluations demonstrate that as compared to the state-of-the-art solutions, PHOENIX can reduce the DoD by up to 54.8% and prolong the battery lifetime to $2.9\times$ while guaranteeing task deadline.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (No. 2022YFB3105202) and National Natural Science Foundation of China (NSFC No. 62132004 and No. 62372259). Qian Wu is the corresponding author.

REFERENCES

[1] G. Yang, J. Lei, W. Xie, Z. Fang, Y. Li, J. Wang, and X. Zhang, "Algorithm/hardware codesign for real-time on-satellite cnn-based ship detection in sar imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–18, 2022.

[2] P. K. Buttar and M. K. Sachan, "Semantic segmentation of clouds in satellite images based on u-net++ architecture and attention mechanism," *Expert Systems with Applications*, vol. 209, p. 118380, 2022.

[3] Y. Cheng, C. Wei, S. Sun, B. You, and Y. Zhao, "An leo constellation early warning system decision-making method based on hierarchical reinforcement learning," *Sensors*, vol. 23, no. 4, 2023.

[4] B. Zhang, Y. Wu, B. Zhao, J. Chanussot, D. Hong, J. Yao, and L. Gao, "Progress and challenges in intelligent remote sensing satellite systems," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 1814–1822, 2022.

[5] A. Russo and G. Lax, "Using artificial intelligence for space challenges: A survey," *Applied Sciences*, vol. 12, no. 10, 2022.

[6] A. A. Osuwa, E. B. Ekhoragbon, and L. T. Fat, "Application of artificial intelligence in internet of things," in *Proceedings of the 9th International Conference on CICN*. IEEE, 2017, pp. 169–173.

[7] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the 25th International Conference on ASPLOS*, 2020, pp. 939–954.

[8] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, "Kodan: Addressing the computational bottleneck in space," in *Proceedings of the 28th International Conference on ASPLOS*, 2023, pp. 392–403.

[9] X. Zhang, J. Liu, R. Zhang, Y. Huang, J. Tong, N. Xin, L. Liu, and Z. Xiong, "Energy-efficient computation peer offloading in satellite edge computing networks," *IEEE Transactions on Mobile Computing*, 2023.

[10] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *the 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.

[11] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[12] D. Vasisht, J. Shenoy, and R. Chandra, "L2D2: Low latency distributed downlink for leo satellites," in *Proceedings of the 2021 ACM SIGCOMM Conference*, 2021, pp. 151–164.

[13] M. Gregory, F. Heine, H. Kämpfner, R. Lange, M. Lutzer, and R. Meyer, "Commercial optical inter-satellite communication at high data rates," *Optical Engineering*, vol. 51, no. 3, pp. 031 202–031 202, 2012.

[14] C. Carrizo, M. Knapek, J. Horwath, D. D. Gonzalez, and P. Cornwell, "Optical inter-satellite link terminals for next generation satellite constellations," in *Free-Space Laser Communications XXXII*, vol. 11272. SPIE, 2020, pp. 8–18.

[15] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen *et al.*, "The ϕ-sat-1 mission: the first on-board deep neural network demonstrator for satellite earth observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.

[16] E. Rapuano, G. Meoni, T. Pacini, G. Dinelli, G. Furano, G. Giuffrida, and L. Fanucci, "An FPGA-based hardware accelerator for CNNs inference on board satellites: benchmarking with Myriad 2-based solution for the CloudScout case study," *Remote Sensing*, vol. 13, no. 8, p. 1518, 2021.

[17] L. Buckley, A. Dunne, G. Furano, and M. Tali, "Radiation test and in orbit performance of MpSoC AI accelerator," in *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 1–9.

[18] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci, "CloudScout: a deep neural network for on-board cloud detection on hyperspectral images," *Remote Sensing*, vol. 12, no. 14, p. 2205, 2020.

[19] S. Wei, X. Zeng, Q. Qu, M. Wang, H. Su, and J. Shi, "HRSID: A high-resolution SAR images dataset for ship detection and instance segmentation," *Ieee Access*, vol. 8, pp. 120 234–120 254, 2020.

[20] D. Spiller, K. Thangavel, S. T. Sasidharan, S. Amici, L. Ansalone, and R. Sabatini, "Wildfire segmentation analysis from edge computing for on-board real-time alerts using hyperspectral imagery," in *2022 IEEE International Conference on MetroXRAINE*. IEEE, 2022, pp. 725–730.

[21] J. Yang, C. Du, W. Liu, T. Wang, L. Yan, Y. Gao, X. Cheng, P. Zuo, Y. Ma, G. Yin *et al.*, "State-of-health estimation for satellite batteries based on the actual operating parameters–health indicator extraction from the discharge curves and state estimation," *Journal of Energy Storage*, vol. 31, p. 101490, 2020.

[22] X. Hu, L. Xu, X. Lin, and M. Pecht, "Battery lifetime prognostics," *Joule*, vol. 4, no. 2, pp. 310–346, 2020.

[23] K. R. Mallon, F. Assadian, and B. Fu, "Analysis of on-board photovoltaics for a battery electric bus and their impact on battery lifespan," *Energies*, vol. 10, no. 7, p. 943, 2017.

[24] J. Li, A. M. Gee, M. Zhang, and W. Yuan, "Analysis of battery lifetime extension in a smes-battery hybrid energy storage system using a novel battery lifetime model," *Energy*, vol. 86, pp. 175–185, 2015.

[25] CelesTrak, "Norad gp element sets," https://celestrak.org/NORAD/elements/, 2023.

[26] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 341–354.

[27] R. M. Nauss, "Solving the generalized assignment problem: An optimizing and heuristic approach," *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 249–266, 2003.

[28] Z. Lai, H. Li, Y. Deng, Q. Wu, J. Liu, Y. Li, J. Li, L. Liu, W. Liu, and J. Wu, "StarryNet: Empowering researchers to evaluate futuristic integrated space and terrestrial networks," in *20th USENIX Symposium on Networked Systems Design and Implementation*, 2023, pp. 1309–1324.

[29] SpaceX, "Application for fixed satellite service by space exploration holdings, llc," https://fcc.report/IBFS/SAT-MOD-2020041700037, 2020.

[30] Oneweb, "Application for fixed satellite service by worldvu satellites limited," https://fcc.report/IBFS/SAT-LOI-20170301-00031, 2017.

[31] L. S. Foundation, "Satnogs network - ground stations," https://network.satnogs.org/stations/, 2023.

[32] Picosats, "Radiosat," https://picosats.eu/wp-content/uploads/2019/02/PICOSATS-RADIOSAT_201811.pdf, 2019.

[33] TESAT, "Cubelct, smallest laser communication transmitter worldwide," https://www.tesat.de/images/tesat/products/220607_DataSheet_CubeLCT100M_A4.pdf, 2021.

[34] D. Rashkovetsky, F. Mauracher, M. Langer, and M. Schmitt, "Wildfire detection from multisensor satellite imagery using deep semantic segmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 7001–7016, 2021.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on CVPR*, 2016, pp. 779–788.

[36] kaggle, "Ships/vessels in aerial images," https://www.kaggle.com/datasets/siddharthkumarsah/ships-in-aerial-images, 2023.

[37] M. Kerr, S. Tonetti, S. Carnara, J. I. Bravo, R. Hinz, A. Latorre, F. Membibre, A. Ramos, S. Wiehle, O. Koudelka *et al.*, "EO-Alert: A satellite architecture for detection and monitoring of extreme events in real time," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. IEEE, 2021, pp. 168–171.

[38] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[39] G. H. d. A. Pereira, A. M. Fusioka, B. T. Nassu, and R. Minetto, "A large-scale dataset for active fire detection/segmentation (landsat-8)," 2020. [Online]. Available: https://dx.doi.org/10.21227/t9gn-y009

[40] Z. Lai, Q. Wu, H. Li, M. Lv, and J. Wu, "OrbitCast: Exploiting mega-constellations for low-latency earth observation," in *the 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–12.

[41] M. Lyu, Q. Wu, Z. Lai, H. Li, Y. Li, and J. Liu, "Falcon: Towards fast and scalable data delivery for emerging earth observation constellations," in *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2023, pp. 1–10.

[42] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM transactions on networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[43] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE transactions on parallel and distributed systems*, vol. 26, no. 12, pp. 3317–3329, 2014.

[44] T. Ren, Z. Hu, H. He, J. Niu, and X. Liu, "FEAT: Towards fast environment-adaptive task offloading and power allocation in mec," in *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2023, pp. 1–10.

[45] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 2020.