Energy and Time-Aware Inference Offloading for DNN-based Applications in LEO Satellites

Yijie Chen, Qiyang Zhang, Yiran Zhang, Xiao Ma, Ao Zhou
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunicatons,Beijing, China
{yijiechen;qyzhang;sgwang}@bupt.edu.cn, {yijiechen;qyzhang;sgwang}

Abstract—In recent years, Low Earth Orbit (LEO) satellites have witnessed rapid development, with inference based on Deep Neural Network (DNN) models emerging as the prevailing technology for remote sensing satellite image recognition. However, the substantial computation capability and energy demands of DNN models, coupled with the instability of the satellite-ground link, pose significant challenges, burdening satellites with limited power intake and hindering the timely completion of tasks. Existing approaches, such as transmitting all images to the ground for processing or executing DNN models on the satellite, is unable to effectively address this issue. By exploiting the internal hierarchical structure of DNNs and treating each layer as an independent subtask, we propose a satellite-ground collaborative computation partial offloading approach to address this challenge. We formulate the problem of minimizing the inference task execution time and onboard energy consumption through offloading as an integer linear programming (ILP) model. The complexity in solving the problem arises from the combinatorial explosion in the discrete solution space. To address this, we have designed an improved optimization algorithm based on branch and bound. Simulation results illustrate that, compared to the existing approaches, our algorithm improve the performance by 10%-18%.

Index Terms—Inference offloading, LEO satellite, DNN-based application, Performance

I. INTRODUCTION

With the continuous technological advancements and the growing demand for space exploration, Low Earth orbit (LEO) satellites, exemplified by constellations like SpaceX, OneWeb, and Telesat [1], are rapidly evolving. LEO satellites serve various purposes with Earth observation being a typical one. This involves capturing ground images and using a Deep Neural Network (DNN) model to infer occurrences such as forest fires, terrain, and geomorphic changes [2], [3].

The prevailing approach involves transmitting all images to the ground for processing, utilizing a traditional method known as the "bent-pipe" architecture [4]. However, the capacity of the satellite-ground link is limited, and the connection between satellites and the ground links is often unreliable and periodic, resulting in a significant amount of data remaining that cannot be timely transmitted. An alternative solution involves executing DNN models on the satellite [5]. However, the majority of LEO satellites are equipped with limited computing power due to their small size and reliance on solar energy collection,

making it extremely challenging to run the entire large-scale DNN models on energy-limited satellites [6]. As an alternative, small-scale DNN models are considered, but they may not guarantee the same level of inference accuracy [7].

To tackle this issue, we exploit the hierarchical structure of DNNs and treat each layer as an independent subtask. The input matrices of one layer are derived from the output matrices of the preceding layer. A feasible strategy involves executing certain layers on the LEO satellite while offloading the remaining layers and the intermediate output to the ground. Considering that the input matrices of most layers in the DNN model tend to decrease compared to the initial input, this strategy effectively utilizes the limited resources of LEO satellites to significantly reduce the transmitted data size.

However, accurately selecting the appropriate layers for offloading is a challenging task due to the following factors: 1) notable variations exist in the internal structure among different DNNs; 2) due to the diverse nature of DNN layers, different offloading strategies yield diverse performances; 3) the constrained computation capability, link capacity, and energy of LEO satellites all contributes to the execution time and onboard energy consumption, making it a coupled and complex problem. To overcome these challenges, we establish a general model to ensure more generalizable results and propose an energy and time-aware inference offloading algorithm to address this problem.

The main contributions of this paper are as follows.

- To the best of our knowledge, we are the first to investigate inference offloading based on the layered architecture of the DNN model in LEO satellites.
- We formulate the problem as a constrained integer linear programming (ILP) problem and propose an improved algorithm based on branch and bound to address it.
- We evaluate the performance of the proposed algorithm through simulations, demonstrating promising experimental results that can support energy-efficient and time-saving offloading.

II. BACKGROUND AND MOTIVATION

"bent pipe" architecture. Most current LEO satellites still employ a communication model known as "bent pipe" architecture. In this architecture, satellites serve as data-transmitting relays without data processing capabilities. In par-
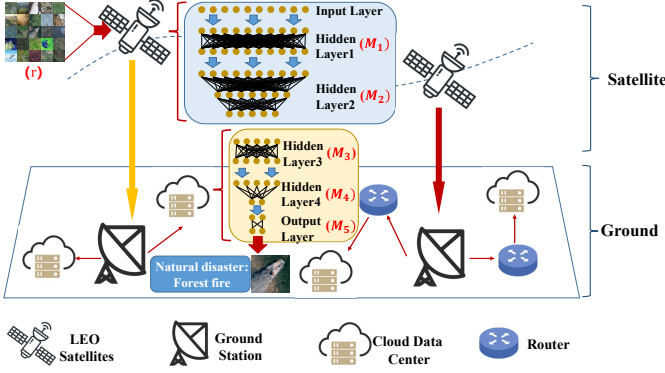
Fig. 1: Architecture of Satellite-Ground Collaborative Computing.

ticular, Cheng et al. [8] proposed a satellite-ground integrated network edge computing architecture that utilizes satellites to enable access to cloud computing resources. Giuliari et al. [9] introduced a kind of inter-networking approach, in which ground stations work as access points or gateways for satellite networks, in order to improve the efficiency of satellite data transmission.

Nonetheless, the data transmission rate from the satellite to the ground is significantly hindered by an unstable communication link and intermittent availability, resulting in a much lower transmission rate compared to the data generation rate on the satellite. Consequently, this led to a substantial increase in the overall latency of entire tasks.

Satellite edge computing. Satellite edge computing has been developed to facilitate on-orbit processing. Xu et al. [10] presented a space-ground-sea integrated network architecture, utilizing satellites as computational nodes for relevant tasks, with the aim of minimizing user execution latency. Denby et al. [5] proposed an Orbital edge computing system that performs partial processing and data filtering on satellites. This system aims to enhance downlink connections' efficiency in saturated satellites while considering energy limitations and payload impacts. In the existing satellite edge computing schemes, the entire inference process is executed on the satellite. This approach imposes a heavy burden on the satellite due to its limited computing resources and low energy acquisition capabilities.

To address the above issues, We consider the varying computational requirements and data volume at different layers of DNNs and investigate a computation offloading strategy aimed at reducing energy consumption and latency.

III. SYSTEM MODEL AND PROBLEM DEFINITION

In this section, we first introduce the system model, context, and notation used in this work. We then provide a more detailed description of the problem being considered.

A. System Model

Our satellite-ground collaborative architecture consists of LEO satellites, ground stations, and cloud data centers, as shown in Fig. 1. Every LEO satellite is equipped with computationally and storage-capable payloads. This enables edge-like satellites to process partial images and serve as data transmission routers. When a satellite comes within the connected range of a ground station, it becomes capable of transmitting data to the ground station. Cloud data centers offer substantial computational power, with some of them directly attached to ground stations, while others are located at a significant distance from the ground station. We denote this architecture as $G = (S \cup DC \cup GS \cup L)$, where S denotes the set of low earth orbit satellites, DC denotes the set of cloud data centers, GS denotes the set of ground stations, and L denotes the set of links among LEO satellites, ground station services, and cloud data centers, respectively.

B. Inference Offloading and Partitioning

Considering the characteristics of DNN models, task partitioning can be used to effectively alleviate the workload on satellites and minimize data transmission. DNN models typically consist of various layers, including input, hidden, convolutional, pooling, and output layers, etc [11], [12]. Each layer requires specific computing resources and incurs energy consumption. Taking advantage of this insight, a large-scale DNN task can be divided into subtasks corresponding to different layers. Notably, convolutional layers employ smaller kernels for feature extraction, while pooling layers reduce the spatial dimensions of feature maps through aggregation and downsampling. Consequently, as the DNN network advances through its layers, the size of feature maps gradually decreases. Considering the limited computational payload of satellites, it is feasible to complete the DNN processing on the satellite up to a specified layer before offloading the remaining data and parameters to the ground. This approach substantially reduces data transmission and processing on the satellite.

To ensure the generalizability of our solution across different DNN models, we adopted the following settings. Given the diverse requirements and varying layer structures of DNN models for different tasks, we didn't concentrate on specific DNNs. Furthermore, numerous excellent experiments have already performed the partitioning of DNNs into subtasks based on layers [13], [14]. Therefore, our focus is on the offloading process after partitioning a DNN model into distinct subtasks. The objective is to determine which layers of the DNN should be processed onboard the satellite and which layers should be offloaded for ground processing. Accordingly, we denote r as a request for a DNN model based on its layer structure. We divide r into $\{M_1, ... M_k, ... M_K\}$, where K is a positive integer and $1 \le k \le K$, M_k represents the subtasks of request r. Let D be the original data size of request r.

C. Latency

An inference request for DNN models can be partitioned into K tasks. The overall latency is determined by the latency

of each task. The latency can be divided into data processing and data transmission latency.

Data processing latency. The latency of a task is influenced by the computing resources available in the satellite and the cloud data center assigned to this task, as well as the data size being processed. In line with our previously established settings, D represents the original data size of request r. Additionally, the input matrix ratio of each layer is typically bounded and predetermined, denoted as α_k. Consequently, the data size of each layer can be expressed as $\alpha_k \cdot D$. And the latency for processing M_k in the satellite S_i can be expressed as follows [15], [16], [17]:

$$\delta_{i,k} = (\alpha_k \cdot D) \cdot \beta_i \tag{1}$$

where β_i is the latency for processing a unit amount of data of S_i, and the processing rate of different LEO satellites varies depending on their payloads and operating environment. Similarly, the latency for processing M_k on the cloud data center can be expressed as follows:

$$\delta_k^{'} = (\alpha_k \cdot D) \cdot \gamma \tag{2}$$

where γ represents the latency for processing a unit amount of data in the cloud data center.

Data transmission latency. Due to the long data transmission time, it is important to consider the latency associated with data transmission. When the satellite transmits data to the ground station, multiple factors come into play and need to be considered. Firstly, the satellite has a limited duration of contact with the ground station during each orbital period, typically around six minutes. Additionally, the connection between the satellite and the ground station undergoes dynamic changes, resulting in a relatively low transmission rate, usually less than 100 Mbps. Consequently, it is crucial to account for situations where the satellite cannot complete the transmission of all the required data within a single transmission cycle. Considering these factors, the latency of transmitting data from the satellite to the ground can be divided into two parts: the latency of satellite data transmission and the latency of waiting for data transmission when the ground station is out of contact with the satellite during the operational cycle. Accordingly, the latency for transmitting the input data of M_k from the satellite to the ground can be expressed as follows:

$$t_k^{'} = t_{tr}^{'} + t_{per}^{'} = \frac{\alpha_k \cdot D}{R_i} + t_{cyc} \cdot (\lceil \frac{\alpha_k \cdot D}{R_i \cdot t_{con}} \rceil - 1) \tag{3}$$

where $t_{tr}^{'}$ represents the transmission time for the input data of M_k, while $t_{per}^{'}$ denotes the waiting time during the transmission of M_k. R_i corresponds to the transmission rate from Satellite S_i to the ground station. Additionally, t_{cyc} is the contact period time between the satellite and the ground station, and t_{con} indicates the duration of communication between a satellite and a ground station.

In scenarios where the data is transmitted to a ground station without an associated cloud data center, the data must be further transmitted to a cloud data center located at a certain distance for processing. In this case, the transmission delay

of M_k from ground station p to cloud data center q can be expressed as follows:

$$t_{g,c} = \frac{\alpha_k \cdot D}{R_{g_p,c_q}} \tag{4}$$

where R_{g_p,c_q} represents the data rate through the channel from ground station p to cloud data center q.

The overall task completion time of task r comprises four components: the execution time on the satellite, the data transmission time from the satellite to the ground station, the data transmission time from the ground station to the cloud data center, and the subsequent task processing time in the cloud data center. Let h_k be a binary variable that indicates whether M_k is executed on the satellite, where $h_k = 1$ indicates that M_k is executed on the satellite; otherwise $h_k = 0$ indicates that M_k is executed on the cloud data center. And $(h_{k-1} - h_k)$ indicates the layer from which data needs to be offloaded from the satellite to the ground. Thus, the total latency of r is given by:

$$\begin{aligned} T =& T_{Satellite} + T_{StoG} + T_{GtoC} + T_{Cloud} \\ =& \sum_{k=1}^{K} h_k \cdot \delta_{i,k} + \sum_{k=1}^{K} (h_{k-1} - h_k) \cdot t_k^{'} \\ &+ \sum_{k=1}^{K} (h_{k-1} - h_k) \cdot t_{g,c} + \sum_{k=1}^{K} (1 - h_k) \cdot \delta_k^{'} \end{aligned} \tag{5}$$

D. Energy Consumption

The total energy consumed by task r is the sum of the energy consumed by each subtask. However, the satellite's computational power is constrained by its payload capacity, limited heat dissipation capabilities in space, and the low energy acquisition rate of solar panels. Consequently, the energy resources of satellites are severely restricted. Conversely, cloud data centers have an abundant electricity supply. Therefore, from a practical perspective, our primary focus is on conserving energy on the satellite. We further divide the energy consumption on the satellite into two components: during data processing and data transmission.

Energy consumption during data processing. Previous research studies [18] and [19] have demonstrated that the energy consumption of a processing unit is directly proportional to both the task access rate and its maximum power consumption. Therefore, our focus centers on the energy consumption model of DNN processing units. As a result, the energy consumption associated with executing M_k of r in S_i can be expressed as follows:

$$e_{i,k}^{satellite} = \delta_{i,k} \cdot (\frac{\alpha_k \cdot D}{\zeta_i \cdot \delta_{i,k}} P_i^{max} + P_i^{idel} + P_i^{leak}) \tag{6}$$

where $\delta_{i,k}$ represents the latency for processing M_k in S_i, ζ_i indicates the maximum amount of data that can be processed per unit time with maximum power, P_i^{max} denotes the maximum power consumption of all GPU units on satellite i, P_i^{idle} represents the idle power consumption of satellite i, and P_i^{leak} refers to the power leakage of the GPU units on satellite i.

Energy consumption during data transmission. Recall that t'_{tr} denotes the transmission time for the input data of M_k, while P_i^{off} represents the data transmission power of the antenna satellite i. The energy consumption of satellite i during the transmission of the input data of M_k (or the output data of M_{k-1}) to the ground can be expressed as:

$$e_{i,k}^{off} = t'_{tr} \cdot P_i^{off} = \frac{\alpha_k \cdot D}{R_i} \cdot P_i^{off} \tag{7}$$

Based on the aforementioned analysis, the comprehensive energy consumption during the execution of a DNN inference task can be formulated as follows:

$$E = \sum_{k=1}^{K} h_k \cdot e_{i,k}^{satellite} + \sum_{k=1}^{K} (h_{k-1} - h_k) \cdot e_{i,k}^{off} \tag{8}$$

E. Problem Formulation

The objective of this model is to effectively address the optimization problem of minimizing energy consumption and reducing task latency. This is achieved through the implementation of efficient partitioning and offloading techniques for DNN models throughout the entire DNN inference process. Considering the diverse nature of satellite tasks, certain specific tasks require meeting low-latency requirements, especially critical applications like fire hazard detection. Conversely, for longer-duration detection tasks, such as remote sensing observations of changes in terrain and landforms, conserving valuable energy resources on the satellite becomes of utmost importance. Considering the varying importance of energy and latency in different applications, we incorporate both factors into the optimization objective by applying appropriate weights. Furthermore, to account for the disparate magnitudes of energy and latency and to ensure that both factors are effectively reflected in the objective function, normalization is applied to both energy and latency. Consequently, the problem can be formulated as an integer linear problem, expressed as follows:

$$min \quad Z = \mu \frac{E - E_{min}}{E_{max} - E_{min}} + \lambda \frac{T - T_{min}}{T_{max} - T_{min}} \tag{9}$$

s.t.:

$$\gamma \geq \gamma_{max} \tag{10}$$

$$\sum_{k=1}^{K} h_k + \sum_{k=1}^{K} (1 - h_k) = K \tag{11}$$

$$\sum_{k=1}^{K} (h_k - h_1) \leq 1 \tag{12}$$

$$h_k \geq h_{k+1} \tag{13}$$

$$h_k \in \{0, 1\} \tag{14}$$

where μ and λ are weighting coefficients, $\mu + \lambda = 1$. And Z is the optimization objective. Eq. (10) specifies the upper limit on the latency for processing a unit amount of data in a cloud data center. Eq. (11) guarantees the integrity of request r by ensuring that all subtasks M_k are executed either on the satellite or in the cloud data center. Eq. (12) indicates that when intermediate results of satellite execution need to be transmitted, there should be exactly one. Eq. (13) makes sure that the data before the down transmission is running on the satellite, and the data after the down transmission is running on the ground. Eq. (12) and Eq. (13) maintain the continuity of request r. Eq. (14) restricts the binary variables.

The problem is an integer linear programming problem with integer constraints, resulting in a combinatorial explosion in the discrete solution space. So we need an algorithm to reduce complexity while obtaining an approximate optimal solution.

IV. INFERENCE OFFLOADING ALGORITHM

We exploit the branch and bound technique to tackle the integer linear programming problem. The branch and bound method continuously partitions the problem space and applies bounding conditions to each subproblem to discovery the optimal solution or approximation optimal solution. Despite being an exhaustive search algorithm, it efficiently reduces the search space by intelligently pruning unnecessary branches. The key idea involves expanding the feasible solution space of the problem as a branching tree and searching for the optimal solution within each branch.

The detailed algorithm is given in Algorithm 1, which is referred to as integer linear programming based on branch and bound method (ILPB). First, initialize the decision variables H, and set the upper bound of the optimal solution (lines 1-3). Next, call the ILPB function and return the offloading optimal decision (lines 4-5). The remaining part is the definition of the ILPB function. When the recursion boundary is reached, return the result (lines 7-9). If the current H satisfies the constraint conditions and is a feasible solution, evaluate the solution. If the current solution Z is better than the optimal solution Ans, update the optimal solution to the current solution; otherwise, Terminate this condition and return. (lines 10-17). Select an undecided integer variable h_k, and for each possible value of the variable, if the current function value plus the minimum possible value of the remaining variables is less than the optimal solution Ans, update the set of decision variables and the optimal solution. Call the ILPB function in the current situation (lines 18-26). This approach effectively reduces the computational complexity by eliminating the consideration of numerous subsets. The presence of integer constraints in the problem results in a combinatorial explosion within the search space.

V. EXPERIMENTS

In this section, we evaluate the performance of our proposed algorithm by comparing it with the following two algorithms:

- **All tasks are offloaded to the ground (ARG):** The satellite transmits all captured data to the ground station, which then transfers them to the cloud data center for centralized processing.

Algorithm 1: Inference Offloading Algorithm

Input: $G = (S \cup DC \cup GS \cup L)$, a sequence of tasks $M_1, ... M_k, ..., M_K$ that belong to r, which need to operation in the limited time. The weighting coefficients: λ and μ

Output: An offloading decision h_k for the inference request.

---

1   $H = \{h_1, ... h_k, ... h_K\} = \{0\}$;
   // Initial offloading state
2   $Cons \leftarrow Constrains(10) - (14)$;
3   $Ans \leftarrow inf$;
4   ILPB$(H, Cons, Ans)$;
   // Call ILPB function
5   Return best offloading decision H;
6   **Function** ILPB$(\{h_k\}, Cons, Ans)$:
7   **if** $|Ans' - Ans| < 1e - 5$ **then**
8   return;
9   **end**
     // Recursive Termination Condition
10   **if** *all Cons satisfied* **then**
11   **if** $Z < Ans$ **then**
12        $Ans \leftarrow Z$;
        // Update the optimal solution
13   update H;
        // Update the offloading solution
14   **end**
15   **else**
16   return;
17   **end**
18   **foreach** *undetermined* h_k **in** H **do**
19   **foreach** *possible value of* h_k **do**
20   **if** $Z(h_k) + minZ(\{\overline{h_k}\}) < Ans$ **then**
21   update H;
22          $Ans \leftarrow Z(h_k) + minZ(\{\overline{h_k}\})$;
23          $Ans' = $ ILPB$(H, Z_{\text{Min}}, Cons, Ans)$;
          // Recursively call the ILPB function to the next state
24   **end**
25   **end**
26   **end**

- **All tasks are completed on the satellite (ARS):** The satellite autonomously executes the entire DNN task on its onboard payload.

A. Experiment Setup

In the simulation, we utilize the experiment parameters from a real-world LEO satellite constellation named "Tiansuan Constellation"[1]. Satellites of this constellation operate at an orbital altitude of approximately 500 km and pass over a

---

[1] http://www.tiansuan.org.cn/

---

ground station for data transmission every 8 hours. The data transmission duration per pass is approximately 6 minutes, and the transmission rate fluctuates within the range of [10, 100] Mbps. The parameters β_i and γ_j on processing unit amount (1 KB) of data in the satellite and cloud data center are varied from ranges [0.01, 0.03] seconds and [0.0001, 0.001] seconds, respectively. The parameter α_j is varied from 0.05^k to 0.9^k, considering the reduction in the size of the input data for a layer compared to the input data of the previous layer. The max processing powers of a satellite are set to [1, 10] Watt [20]. And the size of input computation task is set to [1,1000] GB. Due to the large values of energy and time consumption, we represent the results after applying a logarithmic transformation.

B. Experiment Results

As illustrated in Fig. 2, the energy and time consumption of the three algorithms varies with different initial data sizes. As expected, all three algorithms demonstrate increased energy and time consumption with a larger initial data size. The ILPB algorithm consistently outperforms others in terms of energy and time consumption. Moreover, it exhibits a slower growth rate as the initial data size increases. Our method achieves a significant reduction in overall time and energy consumption, amounting to just 10%-18% of the average values obtained from ARG plus ARS.

Next, we study the energy and time consumption associated with the IPLB, ARG, and ARS algorithms under varying data transmission rates between the satellite and the ground station. As illustrated in Fig. 3, the transmission rate is ranged from 10 to 100 MB/s with a step size of 10. The results demonstrate that the proposed IPLB algorithm consistently achieves lower energy consumption and latency compared to the ARG and ARS algorithms. Moreover, as the data transmission rate between the satellite and the ground increases, both the IPLB and ARG demonstrate decreased total time and energy consumption required on the satellite. However, the energy consumption of the ARS method remains largely unaffected by the data transmission rate due to its exclusive execution of tasks on the satellite.

Finally, we study the energy and time consumption of the IPLB, ARG, and ARS algorithms across various proportions of time and energy, as illustrated in Fig. 4. When the ratio between λ and μ is set to 1:0, indicating that energy consumption is not considered, both the ILPB and ARG algorithms demonstrate comparable total time delays for task execution. Notably, the total time of both the ILPB and ARG algorithms is lower than that of the ARS algorithm. However, when the ratio between λ and μ is set to 0:1, the ILPB algorithm surpasses the ARG algorithm by a substantial margin, indicating its superiority in energy consumption. Moreover, as the proportion of μ increases, the ILPB algorithm significantly reduces energy consumption while maintaining shorter time delays, thereby conserving valuable energy resources on the satellite.
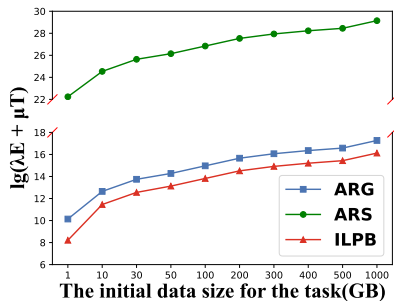
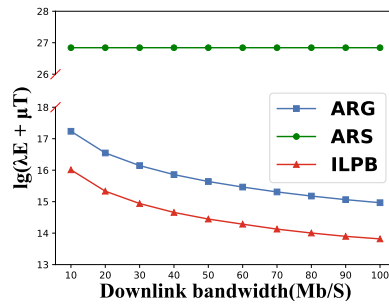Fig. 2: The total consumption of tasks in different initial data size.

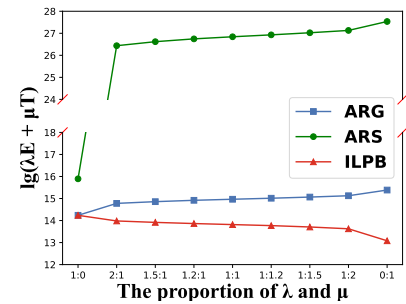Fig. 3: The total consumption of tasks in different transmission rate.

Fig. 4: The total consumption of tasks in different portion for time and energy.

VI. CONCLUSION

This paper addresses the inference offloading problem via satellite-ground collaborative computing, aiming to address challenges associated with limited energy acquisition, insufficient computational resources, and poor satellite-ground communication. We utilize the hierarchical structure of DNNs to partition tasks into subtasks and establish an inference offloading scheme based on a comprehensive metric that encompasses both delay and energy consumption. We incorporate the branch and bound method into the ILP framework to achieve efficient problem-solving. Simulation results validate the effectiveness of the proposed ILPB algorithm in reducing both time and delay. Under different initial data sizes and transmission rates, the algorithm proposed in this paper outperforms the other two strategies. We believe that our approach delivers valuable information and provides an excellent tool that facilitates a more detailed exploration of the inference tasks of LEO satellites. The future work can reduce task complexity through some model lightweight techniques.

VII. ACKNOWLEDGEMENT

REFERENCES

[1] I. Del Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," *Acta astronautica*, vol. 159, pp. 123–135, 2019.

[2] W. Liu, G. Ren, R. Yu, S. Guo, J. Zhu, and L. Zhang, "Image-adaptive yolo for object detection in adverse weather conditions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1792–1800.

[3] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[4] D. Fischer, D. Basin, K. Eckstein, and T. Engel, "Predictable mobile routing for spacecraft networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1174–1187, 2012.

[5] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, "Kodan: Addressing the computational bottleneck in space," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2023, pp. 392–403.

[6] F. Davoli, C. Kourogiorgas, M. Marchese, A. Panagopoulos, and F. Patrone, "Small satellites and cubesats: Survey of structures, architectures, and protocols," *International Journal of Satellite Communications and Networking*, vol. 37, no. 4, pp. 343–359, 2019.

[7] Z. Lai, H. Li, Q. Wu, Q. Ni, M. Lv, J. Li, J. Wu, J. Liu, and Y. Li, "Integrating space edge computing with terrestrial networks for futuristic 6g pervasive on-demand services."

[8] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for iot applications: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[9] G. Giuliari, T. Klenze, M. Legner, D. Basin, A. Perrig, and A. Singla, "Internet backbones in space," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 1, pp. 25–37, 2020.

[10] F. Xu, F. Yang, C. Zhao, and S. Wu, "Deep reinforcement learning based joint edge resource management in maritime network," *China Communications*, vol. 17, no. 5, pp. 211–222, 2020.

[11] Q. Zhang, X. Li, X. Che, X. Ma, A. Zhou, M. Xu, S. Wang, Y. Ma, and X. Liu, "A comprehensive benchmark of deep learning libraries on mobile devices," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 3298–3307.

[12] Q. Zhang, X. Che, Y. Chen, X. Ma, M. Xu, S. Dustdar, X. Liu, and S. Wang, "A comprehensive deep learning library benchmark and optimal library selection," *IEEE Transactions on Mobile Computing*, 2023.

[13] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1423–1431.

[14] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "Ionn: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proceedings of the ACM symposium on cloud computing*, 2018, pp. 401–411.

[15] 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 of Things Journal*, vol. 6, no. 4, pp. 7011–7024, 2019.

[16] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "Ionn: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proceedings of the ACM symposium on cloud computing*, 2018, pp. 401–411.

[17] Q. Yang, X. Luo, P. Li, T. Miyazaki, and X. Wang, "Computation offloading for fast cnn inference in edge computing," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, 2019, pp. 101–106.

[18] S. Hong and H. Kim, "An integrated gpu power and performance model," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 280–289.

[19] C. Luo and R. Suda, "A performance and energy consumption analytical model for gpu," in *2011 IEEE ninth international conference on dependable, autonomic and secure computing*. IEEE, 2011, pp. 658–665.

[20] C. X. Mavromoustakis, G. Kormentzas, G. Mastorakis, A. Bourdena, E. Pallis, and C. D. Dimitriou, "Joint energy and delay-aware scheme for 5g mobile cognitive radio networks," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 2624–2630.