LETTER

Dynamic Fine-grained Task Splitting and Offloading in Integrated Satellite-Terrestrial Network Based on Multi-branch DDQN

Mubiao YAN[†,††,†††], ***Nonmember***, **Xiaohe HE**[†,††,†††], ***Member***, **Luyao WANG**[†,††,†††], **Wenxin YANG**[†,††,†††], **Yuhang LIUZHANG**[†,††,†††], **Hao ZHOU**[†,††,†††], **Zhuochen XIE**[†,†††*], ***and* Huijie LIU**[†*], ***Nonmembers***

SUMMARY Satellite edge computing has garnered significant attention as a key technology enabling low-latency and ubiquitous services in 6G era. This letter introduces an integrated satellite-terrestrial network architecture supporting fine-grained task splitting for collaborative processing. To address the challenge of high-dimensional action space in the dynamic task splitting and offloading problem, we propose a Multi-branch Double Deep Q Network (MBDDQN) approach, featuring dedicated branch per action dimension, shared representation across branches, and dual-layer masking mechanism for reducing noise induced by redundant actions. Numerical simulations demonstrate that MBDDQN achieves superior balance among metrics including successfully scheduled ratio (SSR), on-time completion ratio (OTCR), and task energy consumption over baseline algorithms.

key words: *Integrated satellite-terrestrial network, edge computing, computation offloading, fine-grained task splitting, deep reinforcement learning*

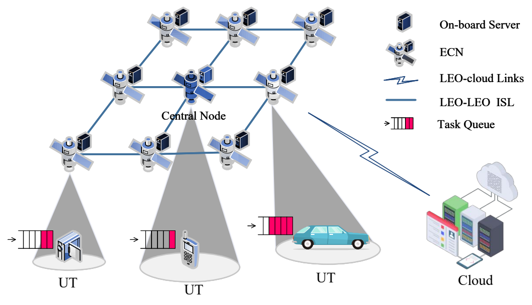


Fig. 1 ISTCCN Scenario

1. Introduction

The exponential growth in global connectivity and service demands has positioned mobile edge computing (MEC) as a critical paradigm, offering low-latency services and enhanced privacy preservation for applications such as autonomous vehicles, Internet-of-Things (IoT), and robotics [1,2]. However, terrestrial MEC faces inherent limitations in coverage due to infrastructure and economic factors. Thanks to the advancements in low-earth orbit (LEO) satellite constellations, satellite-terrestrial integrated network (STIN) has emerged as critical solution to this, while the growing onboard processing capabilities further enable seamless services through satellite edge computing (SEC) [3].

SEC faces significant challenges in task scheduling due to constrained onboard resources while needing to meet diverse task demands. Existing research on SEC has primarily focused on task offloading and resource allocation, where satellites act as edge nodes to process users' tasks either fully or partially. The utility of high-capacity inter-satellite links (ISL) allows cooperative task processing among satellites, enhancing both efficient resource allocation and load balancing [4]. However, collaborative SEC networks must contend with the inherent geographical sparsity and dispersion of onboard resources, leading to uneven resource availability. Meanwhile, prior works have overlooked the considerable variety of data-partitioned oriented tasks [5], for which flexible and finer-grained task splitting could unlock significant potential efficiency gains. While [6] optimized task split-ratio and resource allocation for LEO-assisted IoT networks, their two-part splitting paradigm restricted each task to local/access-satellite processing, preventing multi-way task partitioning and satellite collaboration.

To bridge the gap, this letter presents an Integrated Satellite-Terrestrial Collaborative Computing Network (ISTCCN) supporting fine-grained task splitting, and proposes a multi-branch double deep Q network (MBDDQN) algorithm for joint task splitting and offloading optimization. We formulate the problem as an integer nonlinear programming to minimize communication-computation energy consumption under delay and resource constraints, addressing its high-dimensional action space challenge and redundant actions issue through MBDDQN's shared representation layer, dedicated action branches, and dual-masking mechanism. Numerical simulations verify the efficacy and superiority of MBDDQN in terms of SSR, OTCR, and task energy consumption over benchmarks.

2. System Model and Problem Formulation

2.1 Network Model

As depicted in Fig. 1, we consider an ISTCCN with a three-tier cloud-edge-terminal architecture: (1) terrestrial cloud center node, (2) edge computing nodes (ECNs) denoted as set $\mathcal{S}$, which refers to a constellation consisting of S LEO satellites equipped with edge servers, and (3) user terminals (UTs) denoted as set $\mathcal{U}$ that connect to the nearest LEO satellite to get communication and computing services. LEO

[†]Innovation Academy for Microsatellites of Chinese Academy of Sciences, Shanghai 201304, China.
[††]University of Chinese Academy of Sciences, Beijing 100049, China.
[†††]Key Lab for Satellite Digitalization Technology, Chinese Academy of Sciences, Shanghai 201210, China.
[†]Corresponding authors

satellites are able to communicate with their four neighboring satellites using ISLs. Due to the difference in resources capacity and distances to UTs, the three-tier nodes need to work collaboratively to meet diverse task requirements.

The ISTCCN operates in discrete time slots of duration ρ, with total operation time T partitioned into T_{max} time slots ($\mathcal{T} = \{1, 2, ..., T_{max}\}$). Each UT generates tasks to its queue with length Q within each time slot following Poisson process of arrival rate λ. The kth task of the user u's queue Q_u^p in time slot p is denoted as a four tuple $K_{u,k}^p = \{\tau_{u,k}^p, L_{u,k}^p, c_{u,k}^p, d_{u,k}^{max,p}\}$, which respectively represent the birth time, the data size in bits, the workload in cycles/bit, and the maximum delay tolerance of the task. We assume that the result data of a task is relatively small and the transmission delay of returning it can be neglected [7].

Different from existing works, we consider each task as data-partition-oriented task [5], which can be split evenly into at most G subtasks for parallel offloading to different nodes. This split-and-offload mode provides extra flexibility to the sparse and disperse computing resources in ISTCCN. However, task-splitting results in a G-fold expansion of decision dimension, and the coupling of subtasks complicates the performance analysis, making the problem challenging. The central node of ECNs collects the global information at the beginning of each time slot to make splitting and offloading decisions for all queued tasks. Moreover, ISTCCN adopts a resource reservation scheme to ensure reliable task processing and accommodate the long signaling delay [8].

2.2 Energy and Latency Model for Tasks with Splitting

Given that task $K_{u,k}^p$ is evenly split into $g_{u,k}^p (\leq G)$ subtasks $\{K_{u,k,i}^p\}_{i=1}^{g_{u,k}^p}$, its completion latency can be denoted as:

$$t_{u,k}^p = \rho * (p - \tau_{u,k}^p) + \max_{1 \leq i \leq g_{u,k}^p} t_{u,k,i}^p \ , \tag{1}$$

where $\rho(p - \tau_{u,k}^p)$ is the queuing delay, $t_{u,k,i}^p$ represents the completion latency of $K_{u,k,i}^p$, encompassing both the communication and computation delay:

$$t_{u,k,i}^p = t_{u,k,i}^{\mathrm{comm},p} + t_{u,k,i}^{\mathrm{comp},p}. \tag{2}$$

With queuing energy cost being negligible, the energy consumption of task $K_{u,k}^p$ aggregates its subtask energies:

$$e_{u,k}^p = \sum_{i=1}^{g_{u,k}^p} e_{u,k,i}^p = \sum_{i=1}^{g_{u,k}^p} (e_{u,k,i}^{\mathrm{comm},p} + e_{u,k,i}^{\mathrm{comp},p}) \tag{3}$$

The offloading decision of subtasks of $K_{u,k}^p$ is represented by vector $\mathbf{o}_{u,k}^p = (o_{u,k,1}^p, o_{u,k,2}^p, ..., o_{u,k,g_{u,k}^p}^p)$, where each element $o_{u,k,i}^p$ represents the offloading decision of subtask $K_{u,k,i}^p$ and takes one of the following values:

- 0: subtask remains queued (not scheduled).
- $1, 2, ..., S$: subtask is offloaded to ECN for processing.
- $S + 1$: subtask is computed locally.
- $S + 2$: subtask is offloaded to the cloud.

When subtask $K_{u,k,i}^p$ is computed locally, no data transmission is required, leading to $t_{u,k,i}^{\mathrm{comm},p} = 0$ and $e_{u,k,i}^{\mathrm{comm},p} = 0$. Let f_u in cycles/s or Hz denote the local computing resources for each subtask. The subtask delay is $t_{u,k,i}^{p,loc} = t_{u,k,i}^{\mathrm{comp},p} = L_{u,k}^p c_{u,k}^p / (g_{u,k}^p f_u)$. Based on Dynamic Voltage and Frequency Scaling (DVFS) technique [9] and the CPU energy model, the local computing energy can be expressed as $e_{u,k,i}^{\mathrm{comp},p} = \kappa f_u^2 c_{u,k}^p L_{u,k}^p / g_{u,k}^p$, where κ is the CPU energy coefficient, determined by chip architecture and fabrication.

When subtask $K_{u,k,i}^p$ is offloaded to ECN s' via the access LEO satellite s, the computational delay is $t_{u,k,i}^{\mathrm{comp},p} = L_{u,k}^p c_{u,k}^p / (g_{u,k}^p f_{u,k,i}^{s',p})$, where $f_{u,k,i}^{s',p}$ is the computation capacity allocated by s' (equally shared among subtasks on the same ECN). The computing energy cost follows $e_{u,k,i}^{\mathrm{comp},p} = \kappa (f_{u,k,i}^{s',p})^2 c_{u,k}^p L_{u,k}^p / g_{u,k}^p$. The communication delay is incurred by uplink transmission, propagation, and inter-satellite transmission: $t_{u,k,i}^{\mathrm{comm},p} = L_{u,k}^p / (g_{u,k}^p r_{u,k,i}^{s,p}) + 2d_{u,s}/c' + xL_{u,k}^p / (g_{u,k}^p r_{isl}) + 2xd_{sat}/c'$, where $r_{u,k,i}^{s,p}$ is the uplink rate allocated to the subtask by s (equally shared among subtasks uploaded through s), c' is the light speed, $d_{u,s}$ is the UT-satellite distance, x is the hop count along the route from satellite s to s', r_{isl} is the ISL datarate, and d_{sat} is the inter-satellite distance. The communication energy is $e_{u,k,i}^{\mathrm{comm},p} = p_u L_{u,k}^p / (g_{u,k}^p r_{u,k,i}^{s,p}) + xp_s L_{u,k}^p / (g_{u,k}^p r_{isl})$, with p_u and p_s as the user/satellite transmit powers.

When ECNs and UT resources are insufficient, offloading subtask $K_{u,k,i}^p$ to the cloud becomes preferable. This incurs total delay $t_{u,k,i}^{p,cloud} = L_{u,k}^p c_{u,k}^p / (g_{u,k}^p f_c) + L_{u,k}^p / (g_{u,k}^p r_{u,k,i}^{s,p}) + 2d_{u,s}/c' + L_{u,k}^p / (g_{u,k}^p r_{isl,c}) + 2d_{sat}/c'$ and energy consumption $e_{u,k,i}^{p,cloud} = \kappa f_c^2 c_{u,k}^p L_{u,k}^p / g_{u,k}^p + p_u L_{u,k}^p / (g_{u,k}^p r_{u,k,i}^{s,p}) + p_{s,c} L_{u,k}^p / (g_{u,k}^p r_{isl,c})$, where f_c is the cloud computation capacity. $r_{isl,c}$ represents the ECN-cloud link rate allocated to each subtask. $p_{s,c}$ is the satellite transmit power for ECN-cloud communication.

2.3 Problem Formulation

Aiming to minimize the total energy consumption along the time horizon, we formulate the joint task splitting and offloading (JTSO) problem as follows:

$$\min_{\{g_{u,k}^p\}, \{o_{u,k,i}^p\}} E = \sum_{p \in \mathcal{T}} \sum_{u \in \mathcal{U}} \sum_{k \in Q_u^p} e_{u,k}^p \tag{4}$$

$$\begin{aligned} \text{s.t.} \quad & C1 : t_{u,k}^p \leq d_{u,k}^{max,p}, \quad \forall u, k, p \\ & C2 : o_{u,k,i}^p \in \Omega, \ \forall i \in \{1, 2, ..., g_{u,k}^p\}, \ \forall u, k, p \\ & C3 : g_{u,k}^p \leq G, g_{u,k}^p \in \mathbb{Z}^+, G \in \mathbb{Z}^+, \ \forall u, k, p \\ & C4 : f_{u,k,i}^{s',p} = \frac{F_{s',ava}^p}{|\Psi_{s'}^p|}, r_{u,k,i}^{s,p} = \frac{R_{s,ava}^p}{|\Phi_s^p|}, \quad s, s' \in \mathcal{S}, \forall p \end{aligned}$$

where $\Omega = \{0, 1, 2, ..., S + 2\}$. C4 enforces equal sharing of satellite resources. $\Psi_{s'}^p$ and Φ_s^p denote the sets of subtasks offloaded to satellite s' and to ECNs/cloud via access satellite s in time slot p. $F_{s',ava}^p$ and $R_{s,ava}^p$ represent the available computing and communication resources of satellite s' and s in time slot p, respectively.

3. Dynamic splitting and offloading based on Multi-Branch Double Deep Q Network

The JSTO problem (4) is an integer nonlinear programming with long-term objective and dynamics incurred by stochastic task arrivals and available resources, which is hard to be solved using traditional optimization approaches. Therefore, we reformulate the JSTO problem as a Markov Decision Process (MDP) defined by tuple (S, A, P, R, γ).

(1) State space S: The environment state in time slot p is defined as $s_p = \{Q^p, \boldsymbol{F}^p_{ava}, \boldsymbol{R}^p_{ava}, \boldsymbol{\Gamma}^p\}$, where $Q^p = \{Q^p_1, Q^p_2, ..., Q_{\mathcal{U}}{}^p\}$. Zero-padding is applied to non-full queues to ensure dimensional consistency. $\boldsymbol{F}^p_{ava}$ and $\boldsymbol{R}^p_{ava}$ are the current available computing and communication resources of all ECNs respectively, and $\boldsymbol{\Gamma}^p$ represents the network connection information of ISTCCN.

(2) Action space A: The action space corresponds to the decision variables: $\mathbf{a}^p = \{\{g^p_{u,k}\}, \{o^p_{u,k,i}\}\}_{\forall u,k,i}$. For a given task $K^p_{u,k}$, there are invalid split-and-offload action combinations including the following cases: a) splitting with some of subtasks unscheduled; b) offloading to overloaded nodes; c) offloading to ECNs/cloud when the uplink rate of access satellite is insufficient.

(3) State transition probability matrix P: With each element expressing as $\mathrm{P}(s_p = s' | s_{p-1} = s, \mathbf{a}^{p-1})$, P depicts the intrinsic stochastic dynamics of the environment.

(4) Reward function R: The reward after taking action is defined as the utility of each task: $R^p = \sum\limits_u \sum\limits_k R^p_{u,k}$, where

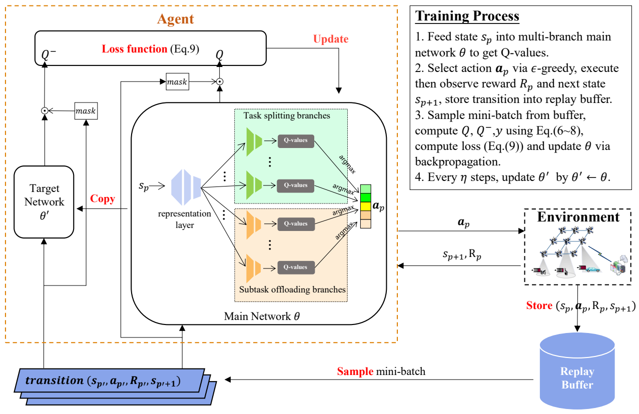
$$R^p_{u,k} = \begin{cases} \mu, & if\ \ \mathbf{a}^p_{u,k} \text{ is invalid} \\ 0, & if\ \ \mathbf{a}^p_{u,k} \text{ is valid } \& \ \mathbf{o}^p_{u,k} = \mathbf{0} \\ r - \omega E - p_1 - p_2, & otherwise. \end{cases} \qquad (5)$$

in which $\mu < 0$ is the penalty for invalid action; $r > 0$ is a positive bias incentivizing task scheduling; ω is the energy cost coefficient; p_1 is the queue overflow punishment for long-term queue stability; p_2 is the delay violation punishment.

(5)Discount factor $\gamma \in [0, 1]$ quantifies the relative importance of future rewards in the computation of the expected long-term return.

The most common Deep Reinforcement Learning (DRL) approach for solving the above discrete-action MDP are value-based algorithms like Deep Q Network (DQN) [10]. However, the standard DQN struggles with the high-dimensional action space induced by split-offload combined decisions and multi-user, multi-task, multi-ECN coordination. Specifically, the action space dimension is $G^{|\mathcal{U}|Q}(S+2)^{|\mathcal{U}|QG}$, which corresponds to the output layer size of Q-network. As the number of total tasks and splitting options grows, the dimension increases exponentially, rendering traditional DQN intractable. To address this, we propose Multi-Branch DDQN (MBDDQN), an extension of Double DQN that alleviates value overestimation of DQN. By designing a shared feature representation and dedicated branch per action component (including $|\mathcal{U}|Q$ task splitting branches and $|\mathcal{U}|QG$ subtask offloading branches, as shown in light colored blocks in Fig. 2), MBDDQN reduces the output layer size from exponential to polynomial $G|\mathcal{U}|Q + (S+2)|\mathcal{U}|QG$, effectively alleviating the curse of dimensionality.

In the training phase, MBDDQN employs both a main



Training Process

1. Feed state s_p into multi-branch main network θ to get Q-values.
2. Select action $\boldsymbol{a}_p$ via ϵ-greedy, execute then observe reward R_p and next state s_{p+1}, store transition into replay buffer.
3. Sample mini-batch from buffer, compute Q, Q^-,y using Eq.(6~8), compute loss (Eq.(9)) and update θ via backpropagation.
4. Every η steps, update θ' by $\theta' \leftarrow \theta$.

Fig. 2 Architecture of the proposed MBDDQN algorithm

network θ and a target network θ', where each branch outputs the Q values for its corresponding action dimension. Experience replay is used in MBDDQN to enhance sample efficiency. For each transition sample, the global Q-value of a state-action pair is derived by summing the branch-specific Q-values, consistent with the additive reward formulation in (5). To handle challenges arising from dynamic queue lengths and variable splitting decisions [9], which create redundant actions and thus extra noise when the network output size is fixed for maximum capacity, we design a queue and split-number aware dual-layer masking mechanism for global Q-value calculation, which is:

$$Q(s, \mathbf{a}) = \sum_u Q_u(s, \mathbf{a}_u) \odot mask(Q_u) \qquad (6)$$

$$\begin{aligned} Q_u(s, \mathbf{a}_u) = \sum_k \Big[& Q_{u,k}(s, g_{u,k}) + \\ & \sum_i (Q_{u,k,i}(s, o_{u,k,i}) \odot mask(g_{u,k}) \Big], \end{aligned} \qquad (7)$$

where $mask(\cdot)$ is a binary mask vector generated by the queue information of user or the split number of task, such that real tasks (non-padding) and subtasks correspond to element 1, and 0 otherwise. Here $\odot$ represents the element-wise multiplication. Accordingly, the Temporal Difference (TD) target is calculated by the Bellman equation [11] as:

$$y = R + \gamma Q^- = R + \gamma Q\big(s', \mathbf{a}' = \{\arg\max_{a'} Q(s', a'; \theta)\}; \theta'\big). \quad (8)$$

in which Q^- is the estimated next-state Q value of the action chosen greedily from the main network. The loss is defined as the mean square of TD error:

$$loss = \mathbb{E}_{(s,\mathbf{a},R,s') \sim \mathcal{D}} \left(y - Q(s, \mathbf{a})\right)^2 \qquad (9)$$

where $\mathcal{D}$ represents the experience replay buffer. Applying gradient backpropagation, the main network θ gets updated. Every η steps, the target network θ' synchronizes with the main network θ. The algorithm architecture and training process of MBDDQN are illustrated in Fig. 2. After training, the main network model can be deployed on the central node to execute online task split-and-offload decisions.

Table 1 Simulation Parameters

Scenario Parameters		Algorithm Parameters	
Parameters	**Value**	**Parameters**	**Value**
Orbit altitude of ECNs	780km	Start learning rate	0.001
Orbit inclination	86.4°	End learning rate	0.00095
ECNs' uplink capacity	200 Mbps	Activation function	ReLU
ECNs' computing capacity	3 GHz	Mini-batch size	256
Cloud computing capacity	10 GHz	Discount factor	0.99
Laser ISL datarate	10 Gbps	Target network update frequency	50
UT computing capacity	200 MHz	Shared representation layer sizes	[512,256]
UT uplink power	2 W	Branch hidden layer sizes	[256,128,64]
Laser ISL transmit power	1000 W	Start greedy factor ϵ	1
ECN-Cloud link power	100 W	End greedy factor ϵ	0.01
Task arrival rate λ	1	ϵ decaying episodes	5000
Task size	10~100 kB	Discount factor γ	0.99
Task workload	1~1.5 kcycles/bit	Training Episodes	10000
Tolerable delay of tasks	0.1~0.5s	Replay buffer size	5000000
Energy coefficient of CPU	10^{-28}		

Table 2 Performance across algorithms

Algorithms	SSR	Average delay(s)	OTCR
Local	1.8%	3.58	0%
Edge-only	60.12%	1.68	1.27%
MBDQN	96.04%	0.8652	13.37%
MBDDQN	96.91%	0.7645	15.12%

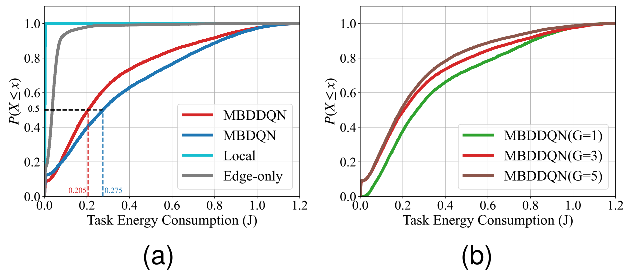


Fig. 3 ECDF of task energy consumption. (a) Comparison among algorithms ($G = 3$) and (b) Comparison under different maximum split number G.

4. Simulation Evaluation and Analysis

We conduct numerical simulations to evaluate our proposed MBDDQN on JTSO problem in ISTCCN scenario. The setup involves 3 UTs with queue length Q=5 at different locations simultaneously requesting services, and 9 neighboring LEO satellites above the vast area of interest from an Iridium LEO constellation serving as the ECNs (Fig. 1). We set T_{max}=200, ρ=0.1s, and maximum number of task splitting G=3. Reward parameters in (5) are configured as: μ=-5, r=5, ω=5, and p_1=p_2=1. Both the representation layer and action branches use fully connected network. Following the typical IoT data processing and low-resolution image recognition tasks requirements [6, 8, 12, 13], additional simulation parameters are listed in Table 1.

We compare the proposed MBDDQN with three baseline algorithms as follows: **(1) Local**, where tasks remain unsplit ($g^{p}_{u,k} \equiv 1$) and are either processed locally or unscheduled; **(2)Edge-only**, where tasks are split up to 2 subtasks and each subtask is either unscheduled, processed locally, or offloaded to its access LEO satellite. **(3) Multi-Branch Deep Q Network (MBDQN)**, which retains the multi-branch network architecture but employs DQN [10] instead of DDQN algorithm to update parameters. We implement all algorithms in Python 3.10 with PyTorch 2.5.1. Performance results are averaged over 20 testing episodes to alleviate the effect of environment stochasticity.

Table 2 presents a comparative performance analysis based on three key metrics (1) successfully scheduled ratio (SSR), defined as the fraction of successfully scheduled tasks out of all requested tasks during time horizon; (2) on-time completion ratio (OTCR); and (3) average delay. It can be seen that MBDDQN outperforms all baselines, achieving improvements of at least 0.87% in SSR, 1.75% in OTCR, and reducing average delay by 0.1007 s. These results validate MBDDQN's reliability for latency-sensitive applications. Besides, the progressive performance gains from Local to Edge-only to MB(D)DQN algorithm demonstrate the benefit of ISTCCN's cloud-edge-terminal collaborative computing paradigm, and validate the effectiveness and scalability of our proposed multi-branch architecture.

We present the empirical cumulative density function (ECDF) curve of task energy consumption in Fig. 3. From Fig. 3a, MBDDQN costs 0.205 J/task at the 50th percentile, which is 25.45% less than that of MBDQN. While Local and Edge-only algorithm show less energy consumption due to much lower local computing frequency and fewer communication procedures, they suffer significantly degraded SSR and OTCR performance (Table 2), making the quality of service unacceptable.

Fig. 3b shows MBDDQN's energy consumption under the varying maximum number of task splitting G. The results demonstrate that our proposed MBDDQN scales well with G, and finer task splitting generally yields better energy efficiency by better utilizing the sparse and disperse resources in ISTCCN. However, diminishing returns suggest a complexity-performance trade-off issue as G grows.

5. Conclusion

In this letter, we present an integrated satellite-terrestrial collaborative computing network architecture for multi-user, multi-task processing with fine-grained task splitting, and propose an approach based on MBDDQN to solve the dynamic task splitting and offloading problem with high-dimensional action space. Simulation experiments validate the efficacy of MBDDQN and demonstrate that it achieves a superior balance among SSR, OTCR, and task energy consumption than baseline algorithms. Besides, we analyze the performance gain enabled by fine-grained task splitting. Future research will investigate constrained task splitting under more complex task types, joint optimization with flexible resource allocation, and algorithms for the high-dimensional hybrid action problem.

References

[1] J. Li, J. Li, Y. Shi, H. Lian, and H. Wu, "A multi-agent deep reinforcement learning algorithm for task offloading in future 6g v2x network," IEICE Trans. Inf.& Syst., vol.advpub, p.2024IIP0005, 2024.

[2] J. Mao, W. Liao, C. Tan, H. Liu, and M. Zheng, "Drl-based task offloading and resource allocation strategy for secure v2x networking," IEICE Trans. Commun., pp.1–13, 2025.

[3] R. Zhang, Y. Feng, Y. Yang, X. Li, and H. Li, "Dynamic delay-sensitive observation-data-processing task offloading for satellite edge computing: A fully-decentralized approach," Remote Sensing, vol.16, no.12, p.2184, 2024.

[4] Y. Hu and W. Gong, "An on-orbit task-offloading strategy based on satellite edge computing," Sensors, vol.23, no.9, p.4271, 2023.

[5] S. Zhang, N. Yi, and Y. Ma, "A survey of computation offloading with task types," IEEE Trans. Intell. Transp. Syst., 2024.

[6] Q. Wang, S. Chen, C. Yang, W. Qi, J. Zong, X. Xia, and D. Wang, "Energy-efficient task split and resource allocation in leo-satellite-assisted iot network," IEEE Internet Things J., vol.11, no.21, pp.34519–34527, 2024.

[7] T.X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," IEEE Transactions on Vehicular Technology, vol.68, no.1, pp.856–868, 2018.

[8] H. Zhang, R. Liu, A. Kaushik, and X. Gao, "Satellite edge computing with collaborative computation offloading: An intelligent deep deterministic policy gradient approach," IEEE Internet Things J., vol.10, no.10, pp.9092–9107, 2023.

[9] P. Peng, W. Lin, W. Wu, H. Zhang, S. Peng, Q. Wu, and K. Li, "A survey on computation offloading in edge systems: From the perspective of deep reinforcement learning approaches," Comput. Sci. Rev., vol.53, p.100656, 2024.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," nature, vol.518, no.7540, pp.529–533, 2015.

[11] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," Proc. AAAI Conf. Artif. Intell., 2016.

[12] 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.

[13] Y. Wang, J. Zhang, X. Zhang, P. Wang, and L. Liu, "A computation offloading strategy in satellite terrestrial networks with double edge computing," 2018 IEEE international conference on communication systems (ICCS), pp.450–455, IEEE, 2018.