Distributed Routing and Data Scheduling in IPNs with GNN-based Multi-Agent DRL

Xixuan Zhou, Ruoyu Xu, Xiaojian Tian, Yueyue Zhang, Yu Liang, Xiaoliang Chen, *Senior Member, IEEE*, and Zuqing Zhu, *Fellow, IEEE*

***Abstract*—As deep space exploration missions grow in complexity, efficient data transfer in interplanetary networks (IPNs) becomes paramount. However, the vast distances, limited bandwidth, and dynamic nature of IPNs pose significant challenges for the routing and data scheduling of interplanetary data transfers (IP-DTs). To address these challenges, this work proposes a novel distributed, graph neural network (GNN) based multi-agent deep reinforcement learning (DRL) approach that can jointly optimize the routing and scheduling of IP-DTs. Our proposal is based on the proximal policy optimization (PPO) framework along with the graph attention networks (GATs). We make the DRL agents for IPN nodes in each subnetwork around a celestial body learn and operate independently, for making intelligent routing and scheduling decisions to properly trade off between average end-to-end (E2E) latency and delivery ratio of IP-DTs while ensuring good scalability. Extensive simulations confirm that our proposal handles the routing and scheduling of IP-DTs much better than existing benchmarks. Further, by modifying the interplanetary overlay network (ION) software platform developed by NASA, we build a semi-physical IPN emulator based on Raspberry Pi boards, implement our proposal in it, and conduct experiments with real data transfers between IPN nodes. Experimental results verify that our proposal can work for practical IPNs without causing excessive overheads and prove its advantages.**

***Index Terms*—Interplanetary networks, deep reinforcement learning, graph attention networks, distributed routing.**

I. Introduction

FUELED by continuous emergence of new technologies, the Internet has been undergoing swift evolution to adapt to the rising demands from both on and around Earth [1–7]. Meanwhile, as humanity constantly explores the mysteries of deep space (DS) for monumental breakthroughs, it becomes crucial to extend the Internet's boundaries to encompass interplanetary networks (IPNs) [8]. This expansion is necessary to accommodate the burgeoning demands for DS explorations whose scopes experience ongoing enlargement [9].

In the universe, celestial bodies engage in a constant dance of motion, making it challenging to realize high-performance data transfer in IPNs. Fig. 1 provides an example of IPN, which covers a five-star system, including Earth, the Moon, Mars, Mercury, and Jupiter. The IPN enables communications among the DS objects, which can be mission operation centers (MOCs), ground stations, satellites, landers, and rovers [10]. Compared with the networks on and around Earth, IPN brings

X. Zhou, X. Tian, X. Chen, and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: xlichen@ieee.org).

R. Xu is with the Eastview Secondary School, Barrie, Ontario, Canada.

Y. Zhang and Y. Liang are with the Shanghai Satellite Network Research Institute and Shanghai Key Lab of Satellite Network, Shanghai 200120, China.

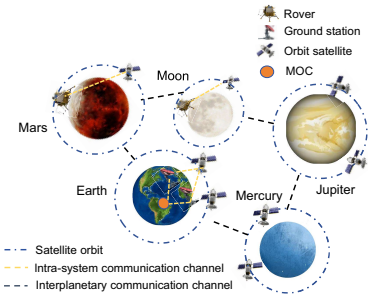


Fig. 1. An example on IPN.

in the following new challenges due to its unique characteristics. First, the vast distances separating DS bodies far exceed those between the network elements on and around Earth. Consequently, IPN exhibits significantly longer link lengths, resulting in substantially long communication delays and restricted data rates. For instance, the average distance between Earth and Mars is about 225 million kilometers, resulting in a one-way transmission delay of $\sim$12.5 minutes [11], and transmission data rate being just a few tens to hundreds of kbps (*e.g.*, $\{8, 32, 128, 256\}$ kbps [12]). Second, the movement of celestial bodies can lead to temporary disruptions in links due to occlusion. The vast network coverage, ultra-long communication delays, and unpredictable link disruptions make IPN topologies dynamic and heterogeneous. While such challenges can be partially addressed by recent innovations in space communications, particularly, delay tolerant networking (DTN) protocols [13–15] based on the store-carry-forward mechanism, they still present significant obstacles for applying centralized network control and management (NC&M).

Hence, it is inevitable to consider distributed routing and data scheduling for IPN. Specifically, to achieve interplanetary data transfers (IP-DTs), we need to divide data into bundles and route and schedule each bundle independently [16]. Previously, researchers have developed a few deterministic algorithms to tackle the distributed routing and data scheduling in IPNs, such as NASA's contact graph routing (CGR) [16], enhancements of CGR [17–20], MARS [21], Lyapunov optimization based approach [22], and fine-grained distributed routing and data scheduling (FD-RDS) [23]. These approaches optimize the routing path of each bundle and realize bundle scheduling in each IPN node to improve the delivery ratio of IP-DTs as well as to reduce their average end-to-end (E2E) latency. Among them, only the studies in [21–23] considered joint optimization of routing and data scheduling of bundles in each IPN node. Nevertheless, the dynamic and heterogeneous nature of IPN makes the joint optimization quite complex,

especially when the network scope scales, and thus the existing deterministic algorithms might not be sufficiently adaptive and can require manual and empirical tuning of key parameters.

On the other hand, recent advances in deep reinforcement learning (DRL) have made it a powerful tool for tackling complex optimization problems in dynamic and heterogeneous environments with minimal human interventions [24]. Therefore, in our preliminary work [25], we proposed to place a DRL agent in each IPN node to achieve distributed routing and scheduling with multi-agent DRL, such that the tradeoff between the delivery ratio and E2E latency of IP-DTs can be properly and automatically balanced. The multi-agent DRL design was based on the asynchronous advantage actor-critic (A3C) framework [26], and its advantages over deterministic policies have been validated in IPNs covering a three-star system (*i.e.*, Earth, Mars and the Moon).

In this paper, we extend our previous work in [25] and present an enhanced multi-agent DRL design for distributed routing and data scheduling of IP-DTs. Specifically, we introduce a DRL-based routing model and a hierarchical data scheduling model, where the upper-level and lower-level DRL agents in each IPN node perform sequential bundle sorting from both macro and micro perspectives, to ensure the scalability of our proposal and its adaptivity to the heterogeneous nature of IPN. The DRL agents are built based on graph neural networks (GNNs), which allows for exploiting the complex topological characteristics of IPNs and potentially generalizing evolving IPN conditions (*e.g.*, topology changes due to link failures). We train the agents with the state-of-the-art proximal policy optimization (PPO) framework [27]. By employing a clipped objective function, PPO has been proved to offer improved stability and sample efficiency, particularly beneficial in our hierarchical multi-agent setting involving time-varying and noisy environment. Performance evaluations through both numerical simulations and experiments verify the effectiveness and superiority of our proposal.

The major contributions over our work in [25] are:

- We devise a novel hierarchical data scheduling model.
- We refine the DRL agent design with GNNs.
- We rearchitect the DRL framework from A3C to PPO, overcoming A3C's defect of prone to unstable training.
- We develop a semi-physical IPN emulator by modifying the interplanetary overlay network (ION) software system [28] to embed our proposal and implementing the software system in Raspberry Pi boards.

The rest of the paper is structured as follows. Section II provides a brief survey of the related work. We elaborate on the network model for distributed routing and scheduling of IP-DTs in Section III. The proposed GNN-based multi-agent DRL-based approach is described in Section IV. Numerical simulations for performance evaluation are discussed in Section V, and in Section VI, we present the implementation of our semi-physical IPN emulator and the experimental results with it. Finally, Section VII concludes the paper.

II. Related Work

Since its inception in late 2000s, the pursuit of effective routing and scheduling solutions for IP-DTs remains an active and ongoing area of research. The pioneering work was done by NASA to define the Bundle Protocol [29] and propose CGR [16] to route bundles for IP-DTs. Specifically, CGR builds a contact graph to represent the potential connection opportunities between IPN nodes and then employs a modified version of Dijkstra's algorithm to plan the routing path of each bundle in the contact graph. However, CGR suffers from a crucial limitation of ignoring the queuing delay of each bundle in an IPN node, and this oversight restricts its scalability to handle heavy traffic loads and IPNs with relatively large topologies. Therefore, researchers have designed several enhanced versions of CGR [17–20] to make routing calculations more accurate with the consideration of queuing delays in IPN nodes. Nevertheless, these CGR enhancements all process bundles in queues in the first-in-first-out (FIFO) manner, without addressing bundle-level scheduling in queues.

The joint optimization of routing and data scheduling of bundles in each IPN node has been tackled in [21–23]. Although they can be more effective after incorporating bundle-level scheduling, these works make use of deterministic algorithms that still lack flexibility and adaptivity to cope with the dynamic and heterogeneous nature of IPN. In particular, MARS [21] relies on manual tuning of system parameters, which significantly restricts its adaptivity and effectiveness in large-scale and complex real-word scenarios. The Lyapunov optimization-based approaches [22] incorporate queue states and link quality in their routing and data scheduling designs but still employ deterministic modeling that can barely adapt to the evolving IPN conditions. To the best of our knowledge, our preliminary work in [25] was the first in the literature that leveraged multi-agent DRL to solve the joint optimization of distributed routing and scheduling of bundles in IPNs. However, as we will show later in Section V, there is still room for further performance improvement. Recently, GNNs have emerged as an effective tool within the machine learning landscape, specifically tailored to process graph-structured data and extract valuable features for applications related to networks. Among GNNs, graph attention networks (GATs) have demonstrated powerful generalization capabilities for network-related optimizations [30].

III. Problem Formulation

In this section, we first explain the network model of IPN and then describe the overall procedure of multi-agent DRL-based distributed routing and scheduling for IP-DTs.

A. Network Model

To capture the time-varying nature of the topology of each IPN, we represent it as a temporal graph $G_t\left(\aleph, \xi_t\right)$, where $\aleph$ is the set of all the IPN nodes and ξ_t denotes the set of temporal links at time t. Here, an IPN can be viewed as a discrete-time system, whose operations rely on distinct time-slots (TS'), each of which lasts for Δt. Hence, the system time becomes $t \in \{0, \Delta t, 2\Delta t, \ldots\}$, which can be normalized as $t \in \mathcal{T} = \{0, 1, 2, \ldots\}$ for simplification [31–34].

As for IP-DTs in an IPN, we still assume that bundles serve as the atomic units [29], and they are transmitted through IPN

nodes with the store-carry-forward (SCF) mechanism, which means that each bundle can be temporarily stored at intermediate nodes as it is transmitted across an IPN, waiting for the next transmission opportunity. The generation of bundles in each IPN node can be modeled by a Poisson process. Each bundle, denoted as B, should reach its destination before a specified deadline, indicating that the current IP-DT has been successfully completed. Following the same paradigm that we modeled in [23], each IPN node $v \in \aleph$ leverages two types of queues: 1) a single queue, denoted as Q_v, acting as the primary buffer for outgoing bundles including bundles generated locally and those traversing v as an intermediate node, and 2) a set of outgoing queues, denoted as $\{Q_{v,u}\}$, each of which stores bundles that need to be transmitted through a specific link $v \rightarrow u$ during future contacts. The performance of routing and data scheduling of bundles can be evaluated with two key metrics, *i.e.*, the delivery ratio and average E2E latency. Table I lists the major notations used in this paper.

TABLE I
MAJOR NOTATIONS

Notation	Description
$v \in \aleph$	A node in the IPN
$e_{v,u} \in \xi_t$	Temporal link $v \rightarrow u$ ($v, u \in \aleph$) at time $t \in \mathcal{T}$
$r_{v,u}$	Capacity of $e_{v,u}$ for $v \rightarrow u$
B_s, B_d, $B_{\text{last-hop}}$	Source, destination and last hop of bundle B
B_{dead}, B_{size}	Deadline and data size of bundle B
B_{ttl}	Current time-to-live of bundle B
B_{proj}	Projected delivery time of bundle B
B_{path}	Current routing path of bundle B
B_{mode}	Flag to tell whether use *Algorithm* 2 on bundle B
Q_v	Local queue on node v to buffer bundles
$Q_{v,u}$	Outgoing queue on node v for $v \rightarrow u$
$L_{v,u}$	Current length of $Q_{v,u}$
$T^a_{v,u}$	Time needed to transfer data buffered in $Q_{v,u}$
$T^B_{v,u}$	Time needed to transfer bundle B through $e_{v,u}$
$T_{v,B}$	Queueing delay of bundle B on node v
$\left\{S^R_B, A^R_B, R^R_B\right\}$	Elements of DRL agents for routing bundle B
$\left\{S^U_{v,u}, A^U_{v,u}, R^U_{v,u}\right\}$	Elements of DRL agents for upper-scheduling in $Q_{v,u}$
$Set_{v,u}$	Set of values $A^U_{v,u}$ of all bundles belonging to $Q_{v,u}$
$C \in \Omega$	A cluster in $Q_{v,u}$
$\left\{S^L_{v,u}, A^L_{v,u}, R^L_{v,u}\right\}$	Elements of DRL agents for lower-scheduling in $Q_{v,u}$

B. Multi-Agent DRL-based Routing and Scheduling

To adapt to the heterogeneous nature of IPN, we categorize the IPN nodes on/around each planet as belonging to a specific planet-centric subnetwork. For example, ground stations located on Earth, and satellites around Earth all belong to the Earth-centric subnetwork. As the IPN nodes in a same subnetwork are relatively close to each other, we assume that the DRL agents on them can share the same experience pool through intra-subnetwork communications. This is because the operations of IPN nodes in a same subnetwork possess certain similarities, and thus making their DRL agents share an experience pool can improve their training efficiency.

Algorithm 1: DRL-based Routing and Data Scheduling

```
1  load topology and contact plan of IPN;
2  initialize local queues and outgoing queues on IPN nodes;
3  for each TS t do
4      refresh G_t(ℵ, ξ_t) according to current contact states;
5      insert newly generated/received bundles in {Q_v};
6      for each node v ∈ ℵ do
7          for each bundle B ∈ Q_v with B_mode = 0 do
8              if B_path = ∅ then
9                  compute B_path with CGR;
10             end
11             fetch next hop u from B_path;
12             move B from Q_v to outgoing queue Q_{v,u};
13         end
14     end
15     for each node v ∈ ℵ do
16         for each bundle B ∈ Q_v with B_mode = 1 do
17             apply Algorithm 2 to get next hop u for B;
18             update B_path;
19             move B from Q_v to outgoing queue Q_{v,u};
20         end
21     end
22     for each link e_{v,u} ∈ ξ_t do
23         if e_{v,u} is available then
24             apply Algorithm 3 to schedule bundles;
25             send bundles as scheduled;
26         else
27             for each bundle B ∈ Q_{u,v} do
28                 set B_mode = 1;
29                 move B from Q_{v,u} back to Q_v;
30             end
31         end
32     end
33     update the TTL values of bundles;
34     remove expired bundles in {Q_v, {Q_{v,u}}};
35     record successful IP-DTs and their E2E delays;
36 end
```

The overall procedure of our proposed multi-agent DRL-based routing and scheduling for IP-DTs is detailed in *Algorithm* 1. The core idea is that at each TS, all the nodes first perform concurrent and local IP-DT routing with CGR or DRL (only for bundles missed their previously-planned transmission windows) and move bundles to the corresponding outgoing queues, and then schedule bundles in each outgoing queue with the two-level DRL. *Lines* 1-2 are for the initialization. Then, the for-loop spanning *Lines* 3-36 iterates through each TS to route and schedule bundles in IPN nodes. In each iteration, *Lines* 4-5 update the IPN's status at the beginning of each TS t. Next, we use two for-loops, covering *Lines* 6-14 and *Lines* 15-21, respectively, to handle the routing and scheduling of IP-DTs in each node, and the for-loop of *Lines* 22-32 schedules bundles in each outgoing queue for a specific link. It can be seen that the operations in different IPN nodes are actually independent, and thus *Algorithm* 1 is a distributed algorithm, allowing for it to potentially work for IPNs in very large sizes.

The for-loop of *Lines* 6-14 iterates through each node $v \in \aleph$ and checks the B_{mode} of each bundle $B \in Q_v$. We define B_{mode} as the flag to indicate whether DRL-based routing, specifically, *Algorithm* 2, should be applied to the corresponding bundle B. By default, B_{mode} is set as 0, indicating that CGR should be used to find the routing path of B, and *Algorithm* 2 is not

necessary. *Lines* 8-10 find a routing path for B if it has not been assigned one yet. Following this, *Lines* 11-12 manage the movement of bundles between local queues and outgoing queues. The subsequent for-loop, spanning *Lines* 15-21, deals with the bundles whose B_{mode} flags were set to 1, *i.e.*, their routing paths should be determined by *Algorithm* 2.

Next, bundle scheduling and transmission for each link is managed by the for-loop of *Lines* 22-32. When a link $e_{v,u}$ is available and suitable for IP-DTs, we sort the bundles in the corresponding outgoing queue $Q_{v,u}$ using a DRL-based data scheduling algorithm (*Algorithm* 3), which is a hierarchical algorithm accomplishing bundle scheduling in three steps: **sorting, clustering, and intra-cluster re-sorting**. Specifically, *Algorithm* 3 (*i*) first sorts to establish a relative order among the bundles with the upper-level DRL data scheduling, (*ii*) then the sorted bundles are clustered using the hierarchical density-based spatial clustering of applications with noise (HDBSCAN) method [35], leveraging the ordering information from the previous sorting stage, and (*iii*) finally, within each cluster, bundles are resorted according to the information of the cluster, with the lower-level data scheduling (DRL-based data scheduling with GNN). This approach can enhance data scheduling and transmission efficiency, particularly in scenarios where a large number of bundles are queued and awaiting forwarding. When a link $e_{v,u}$ is unavailable, we examine each nonempty queue $Q_{v,u}$ at node v, *i.e.*, the queue contains bundles that will miss their transmission windows in the current contact of the link. For these bundles, we set B_{mode} to 1, to trigger rerouting for them in the next TS. Finally, *Lines* 33-35 update IPN status at the end of the current TS.

IV. GNN DRL-based Routing and Data Scheduling

This section delves into the details of our proposed DRL-based algorithms. We first explain the proximal policy optimization (PPO) framework [27] by outlining its key features and describe the principle of GNNs. Then, we elaborate on the design of our proposed routing and hierarchical data scheduling algorithms implemented within this framework.

A. Background of Proximal Policy Optimization (PPO)

Reinforcement learning (RL) enables agents to learn optimal actions to respond to a dynamic environment, such that their cumulative reward can be maximized. DRL integrates deep neural networks into this process to architect agents, helping agents to learn from high-dimensional complex inputs and make more intelligent decisions. As for the training of the DRL agents, we leverage the PPO framework. Empirically, smaller and incremental policy updates during training can promote better convergence towards optimal solutions. Therefore, to ensure stable and reliable learning, PPO implements a mechanism for conservative policy updates, involving calculating a ratio that quantifies the magnitude of policy changes. In particular, a clipped surrogate objective function $L^{\text{CLIP}}(\theta)$ is introduced for limiting the magnitude of policy changes between iterations [27]. Formally, $L^{\text{CLIP}}(\theta)$ is defined as,

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right], \tag{1}$$

where A_t is the advantage function (*i.e.*, $A_t > 0$ means that the current action is the best among those for the current state), $r_t(\theta)$ is the probability ratio between the current and previous policies (see Eq. 2) for measuring the divergence between them, and function $\text{clip}(\cdot)$ clips the value of $r_t(\theta)$ to penalize moving $r_t(\theta)$ outside the interval of $[1-\epsilon, 1+\epsilon]$.

$$r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)}. \tag{2}$$

The combination of the clipped objective function, efficient advantage estimation, on-policy learning, and relative easy implementation ensures the performance of PPO [27].

B. Structures of Graph Neural Networks (GNNs)

We architect the DRL agents based on graph attention networks (GATs) [30], which elevate the capabilities of traditional GNNs by incorporating the attention mechanism. This enables GATs to learn richer representations of nodes in a graph. Unlike traditional GNNs that consider neighboring nodes with equal importance, a GAT intelligently weights the contribution of each neighbor based on its relevance to the target node. This targeted focus on the most informative edges empowers GATs to achieve superior performance in various graph-related tasks.

In a GAT, each node recursively calculates node embeddings taking a feature vector as the initial input. Specifically, the node embedding of layer-$(l+1)$ is obtained from the embedding of layer-l as follows. The first step is,

$$z_i^{(l)} = W^{(l)} h_i^{(l)}, \tag{3}$$

where $h_i^{(l)}$ is the embedding of layer-l and $W^{(l)}$ is its learnable weight matrix. Then, the GAT computes a pair-wise unnormalized attention score between two adjacent nodes by,

$$e_{ij}^{(l)} = \text{LeakyReLU}\left(\vec{\boldsymbol{a}}^{(l)^T}\left(z_i^{(l)} \| z_j^{(l)}\right)\right). \tag{4}$$

Here, the $||$ operator concatenates z embeddings of the two nodes, and the dot product of the concatenated embedding and a learnable weight vector $\boldsymbol{a}^{(l)}$ is fed to the LeakyReLU function. We normalize the attention scores of a node's incoming edges

$$\alpha_{ij}^{(l)} = \frac{\exp\left(e_{ij}^{(l)}\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(e_{ik}^{(l)}\right)}. \tag{5}$$

Finally, each node updates its embedding by aggregating the embeddings from neighbors weighted by the attention scores,

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} \cdot z_j^{(l)}\right), \tag{6}$$

where $\sigma(\cdot)$ is the chosen activation function.

C. DRL-based Routing for IP-DTs

For the routing of IP-DTs, the major drawback of existing deterministic algorithms (*e.g.*, CGR and its enhancements) lies in that they can hardly adjust the routing paths of bundles adaptively, especially after a bundle has missed its scheduled transmission window. To mitigate this limitation, we propose a novel GNN-based DRL routing algorithm that can dynamically

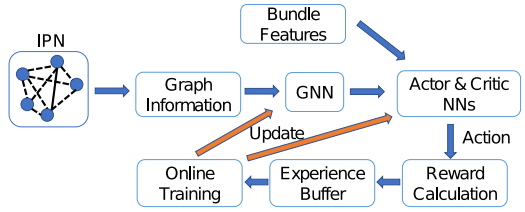


Fig. 2. Architecture and operation principle of our GNN-based DRL model.

update the routing paths for the bundles that have missed their scheduled transmission windows. We place a DRL agent in each IPN node, and then by aggregating available information about the nodes in its subnetwork and its own contact plan using a GNN, the agent is able to make accurate routing decisions based on a broader understanding of the IPN status.

- **Agent:** Each GNN-based DRL agent handles the routing of the bundles that have missed their transmission windows in the previous TS. Specifically, we deploy an Actor neural network (NN) and a Critic NN in each agent, where the former selects an action A_B^R based on S_B^R while the latter evaluates the pros and cons of the chosen action.
- **State:** Each state S_B^R of a bundle B consists of two elements, *i.e.*, the features of B and the IPN status at TS t. The features of bundle B are represented by a vector $B_R = \{B_{\text{size}}, B_{\text{d}}, B_{\text{last-hop}}\}$, and we model the IPN status as graph-structured data $G_t\,(V, E)$. Here, each node $v \in V$ corresponds to an IPN node, and its features are updated according to the current topology, with information about all neighboring nodes in its subnetwork. Specifically, for each neighboring node u, the nearest contact deadline, the length of its outgoing queue $Q_{v,u}$ ($L_{v,u}$), and the time required to transmit all the bundles in $Q_{v,u}$ ($T^a_{v,u}$) are included. Each edge $e \in E$ denotes a specific contact in the current IPN. We put $G_t\,(V, E)$ in the GNN of the agent in node v.
- **Action:** Each action A_B^R determines the bundle's routing path by selecting a suitable and available neighboring node u as the next hop for bundle B.
- **Reward:** The reward R_B^R is obtained after bundle B has been scheduled in the outgoing queue $Q_{v,u}$. We set R_B^R to be proportional to the gap between the original projected delivery time without invoking the DRL-based routing and the new projected delivery time by using the next hop from the DRL-based routing. Note that, the new projected delivery time also includes the queuing delay for B, which is caused by its scheduled order in $Q_{v,u}$ and can be calculated along with *Algorithm* 3.

Algorithm 2 describes the training process of GNN-based DRL for routing IP-DTs, which involves policy calculation and decision making, state update, reward computation, and GNN parameter tuning with PPO by each agent. *Lines* 1-2 initialize the DRL agents, each of which contains a GNN, an Actor-NN, and a Cirtic-NN as shown in Fig. 2. The for-loop of *Lines* 3-26 explains the online training of agents in each TS. In each node v, we first get all the bundles that are in Q_v and need to be rerouted (*Lines* 4-5). The agent in node v leverages its GNN to handle the current status of IPN together with feature vector B_R to get the current state S_B^R. Then, the agent receives S_B^R, puts it into its Actor-NN to get an action A_B^R (*i.e.*, a proper next hop u for bundle B) and moves B from Q_v to $Q_{v,u}$ (*Lines* 6-9). *Lines* 12-25 are the subsequent operations of online training, where *Lines* 15-17 explain how to get R_B^R. After performing hierarchical DRL-based data scheduling on each link $e_{v,u}$, we get $T_{v,B}$ (*i.e.*, the queuing delay of bundle B), and add it to the new projected delivery time to update B_{proj}. Then, the agent computes reward R_B^R, and pushes $\{S_B^R, A_B^R, R_B^R\}$ into its experience buffer as a training sample (*Lines* 17-18). Adhering to the training principle of PPO, we run an iteration of online training after enough samples have been collected.

Algorithm 2: Training of GNN-based DRL for Routing

```
1  initialize parameters of each DRL agent randomly;
2  empty all the experience buffers of DRL agents;
3  for each TS t do
4      for each node v ∈ ℵ do
5          for each bundle B ∈ Q_v with B_mode = 1 do
6              get state S_B^R and input it to corresponding agent;
7              agent outputs a proper action A_B^R;
8              perform A_B^R and get the next hop u;
9              move B to the outgoing queue Q_{v,u};
10         end
11     end
12     for each link e_{v,u} ∈ ξ_t do
13         for each bundle B ∈ Q_{v,u} do
14             perform hierarchical DRL-based data scheduling;
15             if B_mode = 1 then
16                 get T_{v,B} and update B_proj;
17                 agent computes reward R_B^R;
18                 push {S_B^R, A_B^R, R_B^R} into agent's experience
                   buffer as a training sample;
19                 if enough samples have been collected then
20                     agent updates parameters based on PPO;
21                     agent empties its experience buffer;
22                 end
23             end
24         end
25     end
26 end
```

D. Hierarchical DRL-based Data Scheduling for IP-DTs

For the scheduling of bundles in each IPN node, we propose a hierarchical DRL-based data scheduling algorithm that accomplishes data scheduling with the **sorting, clustering, and intra-cluster re-sorting** pipeline, optimizing bundle scheduling from both macro and micro perspectives. Specifically, the sorting step is carried out by the upper-level DRL-based scheduling, followed by the clustering that employs the HDBSCAN [35] method, and finally, the lower-level GNN-based DRL scheduling is responsible for intra-cluster re-sorting.

Both the upper- and lower-level data scheduling algorithms adhere to the following principles, which are derived from the fundamental characteristics of IPNs and the SCF paradigm.

- ▶ Bundles with larger TTL values should be sent out earlier because of their greater chances of arriving at destinations before deadlines.
- ▶ Smaller-sized bundles should be sent out earlier to avoid head-of-line blocking for shorter average queuing delay.

TABLE II
METRICS FOR EVALUATING THE STATUS OF AN OUTGOING QUEUE

Metric	Definition and Formula
Cosine similarity (CS)	It measures the similarity between two multi-dimensional vectors $\mathbf{A}$ and $\mathbf{B}$ as $S_C(\mathbf{A},\mathbf{B}) := \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$.
Coefficient of variation (CV)	It measures the dispersion of a probability distribution as $CV(X) = \frac{\text{mean}(X)}{\text{var}(X)}$, where X is a random variable, mean(X) is its mean value, and var(X) is its standard deviation.
Normalized discounted cumulative gain (NDCG)	It measures the effectiveness of a ranking system by considering the position of each item in the ranked list, and can be used to evaluate the scheduling result for an outgoing queue. We calculate the NDCG value nDCG(Q,Y) of a queue Q in terms of attribute Y of each bundle in it with the method in [36].

► Bundles with earlier projected delivery time should be sent earlier, because it suggests a better routing path with fewer link disruptions.

Meanwhile, we utilize the three metrics listed in Table II to evaluate the status of an outgoing queue.

1) Upper-level DRL-based Data Scheduling: The elements of the upper-level DRL-based data scheduling are as follows.

- **Agent:** A DRL agent is placed in each IPN node to make preliminary decisions regarding the specific order of bundles in each outgoing $Q_{v,u}$.
- **State:** Each state $S^U_{v,u}$ is denoted as a vector. For each bundle B in outgoing queue $Q_{v,u}$, we define $S^U_{v,u}$ as:

$$S^U_{v,u} = \left\{L_{v,u}, \overline{T^a_{v,u}}, \mathbf{B}, \mathbf{C}, S_C(\mathbf{B},\mathbf{B}^*)\right\}, \tag{7}$$

where $L_{v,u}$ represents the queue length of $Q_{v,u}$, and $\overline{T^a_{v,u}}$ denotes the average transmission time of bundles as:

$$\overline{T^a_{v,u}} = \frac{\frac{1}{r_{v,u}} \sum\limits_{B\in Q_{v,u}} B_{\text{size}}}{L_{v,u}}. \tag{8}$$

Here, we have $\mathbf{B} = \{B_{\text{size}}, B_{\text{ttl}}, B_{\text{proj}}\}$ as the attribute vector of B, and $\mathbf{C} = \{C_V(B_{\text{size}}), C_V(B_{\text{ttl}}), C_V(B_{\text{proj}})\}$ as the coefficient of variation (CV) vector of the bundle attributes in $Q_{v,u}$. The function $S_C(\mathbf{B},\mathbf{B}^*)$ in Eq. (7) calculates the cosine similarity (CS) between $\mathbf{B}$ and the ideal normalized vector $\mathbf{B}^* = [1,1,1]$.

- **Action:** Each action $A^U_{v,u}$ is defined as the ranking assigned to bundle B in $Q_{v,u}$, conveyed by a real value between $[0,1]$. A smaller value of $A^U_{v,u}$ signifies that bundle B is scheduled for later transmission, *i.e.*, B is placed at a later position in $Q_{v,u}$. Specifically, when $A^U_{v,u}$ approaches 0, bundle B will be transmitted towards the end of the service time of $Q_{v,u}$, and *vice versa*.
- **Reward:** We define the reward of each action as,

$$\begin{aligned} R^U_{v,u} =& \alpha\cdot \text{nDCG}(Q_{v,u}, B_{\text{size}}) + \beta\cdot \text{nDCG}(Q_{v,u}, B_{\text{ttl}}) \\ &+ \gamma\cdot \text{nDCG}(Q_{v,u}, B_{\text{proj}}) + \mu\cdot e^{-k\cdot T_{v,B}}, \end{aligned} \tag{9}$$

where α, β, and γ are weights determined by the CVs of attributes $\{B_{\text{size}}\}$, $\{B_{\text{ttl}}\}$, and $\{B_{\text{proj}}\}$, respectively, of bundles in $Q_{v,u}$, μ is a positive coefficient, and k is a constant used to rescale $T_{v,B}$, which represents the queuing delay of bundle B after scheduling. The first three terms of Eq. (9) measures the overall performance of the scheduling for $Q_{v,u}$ by applying the normalized discounted cumulative gains (nDCGs) on different bundle attributes (*i.e.*, the size B_{size}, TTL B_{ttl}, and projected delivery time B_{proj}). Here, nDCG assesses how well the scheduling result aligns with the ideal sorting in terms of these attributes. The last term, $\mu\cdot e^{-k\cdot T_{v,B}}$, evaluates the scheduling result of bundle B. As $T_{v,B}$ is the queuing delay of B after scheduling, a smaller value of $T_{v,B}$ leads to a larger contribution to the reward due to the exponential decay function. In all, the reward function considers both the overall scheduling performance of $Q_{v,u}$ and ranking of a bundle in it.

2) Clustering: When there is a high volume of bundles in $Q_{v,u}$, the performance of the upper-level DRL-based scheduling may be compromised due to the presence of similar bundles. This similarity can result in less-than-optimal sorting decisions, highlighting the need for a further sorting process. Since HDBSCAN offers several advantages in clustering tasks, we choose it for the further sorting. Specifically, with HDBSCAN, we cluster the bundles in $Q_{v,u}$ based on the values of $A^U_{v,u}$, and then after the clustering, bundles in $Q_{v,u}$ are divided into several clusters C in set Ω.

3) Lower-level DRL-based Data Scheduling: The lower-level DRL further optimizes the data scheduling by obtaining an effective ordering of bundles in each cluster. The elements of the lower-level DRL-based scheduling are as follows.

- **Agent:** A DRL agent is placed in each IPN node to precisely sort the bundles in a cluster C.
- **State:** Each state $S^L_{v,u}$ contains information about the characteristics in cluster C and features of bundles in it. The information of C is represented by a vector $\mathbf{C_L} = \{L_C, \mathbf{C}\}$, where L_C is the number of items in C and $\mathbf{C} = \{C_V(B_{\text{size}}), C_V(B_{\text{ttl}}), C_V(B_{\text{proj}})\}$ is the CV vector of bundle attributes in C. The details of C is modeled as a graph $G_C(V_C, E_C)$. Each node in G_C symbolizes a bundle in C and its features $\mathbf{B} = \left\{B_s, B_{\text{size}}, B_d, B_{\text{ttl}}, B_{\text{proj}}, S_C(\mathbf{B},\mathbf{B}^*), T^B_{v,u}\right\}$. When the CS value of two bundles exceeds 0.5, a link is inserted to connect the nodes representing them. We put G_C into a GNN to get the output C_L to generate each state $S^L_{v,u}$.
- **Action:** Each action $A^L_{v,u}$ is defined as the ranking of bundle B within its cluster, similar to $A^U_{v,u}$.
- **Reward:** $R^L_{v,u}$ is calculated by the same method for $R^U_{v,u}$, using the information of cluster C after resorting.

4) Hierarchical DRL-based Data Scheduling for IP-DTs: We use PPO for the training of upper- and lower-level DRLs. The training of the DRL for upper-level data scheduling can be done with an approach similar to that used in our previous work [25]. After training of the upper-level agents, we cluster the bundles sorted by them in each outgoing queue, and proceed to train the lower-level agents. The training proce-

dure of GNN-based lower-level scheduling is summarized in *Algorithm* 3, which adopts a state calculation, action, feedback and parameter tuning pipeline that resembles *Algorithm* 2.

Algorithm 3: Training of Lower-level DRL for Scheduling

```
initialize parameters of each DRL agent randomly;
empty all the experience buffers of DRL agents;
for each TS t do
    for each link e_{v,u} ∈ ξ_t do
        for each bundle B ∈ Q_{v,u} do
            perform the upper-level DRL-based scheduling;
        end
        cluster bundles in Q_{v,u} with HDBSCAN method;
        put all the clusters in set Ω;
        for each cluster C ∈ Ω do
            for each bundle B ∈ C do
                get state S^L_{v,u};
                get a proper action A^L_{v,u} according to the
                corresponding upper-level DRL agent;
            end
            sort bundles by A^L_{v,u};
            agent computes reward R^L_{v,u};
            push {S^L_{v,u}, A^L_{v,u}, R^L_{v,u}} into agent's lower-level
            experience buffer as a training sample;
            if enough samples have been collected then
                agent updates its parameters based on PPO;
                agent empties its experience buffer;
            end
        end
    end
end
```

After training the upper-/lower-level agents, we perform hierarchical DRL-based data scheduling for IP-DTs with the procedures in *Algorithm* 4. Specifically, *Algorithm* 4 invokes the upper-level DRL, clustering and lower-level DRL progressively, to optimize the ordering of bundles in the outgoing queues while differentiating similar bundles through two-level inspections. We first set up all the upper-/lower-level DRL agents with trained parameters (*Line* 1). *Lines* 4-7 explain the procedures of the initial ranking with the upper-level agents and how to acquire the information needed for the subsequent clustering. Then, we use HDBSCAN to cluster bundles according to the distribution of their $A^U_{v,u}$ values, and form set Ω to include all the clusters in $Q_{v,u}$ (*Lines* 8-9). We set A^U_C as the median of the $A^U_{v,u}$ values of bundles in each cluster C (*Line* 10). *Lines* 11-15 leverage the lower-level agents to resort bundles in each cluster C. Finally, *Line* 16 implements the overall sorting.

E. Convergence and Complexity of Our Proposal

In Section V, we will demonstrate the training performance through simulations to show that our GNN-based multi-agent DRL proposal converges well, even in large-scale topologies[1]. Especially, it is extremely difficult to establish formal theoretical proofs for the convergence and bounds of our proposed DRL framework in the complex, multi-agent, and hierarchical setting and stochastic IPN environments. Despite these challenges, our proposal indeed shows its efficiency and effectiveness in numerical simulations and experimental demonstrations, as we will explain in Sections V and VI below.

The overall algorithm consists of three main steps: GNN-based routing, upper-level data scheduling, and lower-level data scheduling (GNN-based), all of which are based on the PPO framework. The complexity of the algorithm can be analyzed as follows. The complexity of an L_g-layer GNN can be denoted as $O(L_g(|E|F + |V|F^2))$, where $|V|$ and $|E|$ are the numbers of nodes and edges of graph-structured data $G_t(V, E)$, respectively, and F is the number of features per node. An Actor network with L_a layers and P_a parameters per layer, when processing a batch of N samples, runs in $O(NL_aP_a)$, including both forward and backward propagations. Similarly, a Critic network with L_c layers and P_c parameters per layer, when processing a batch of N samples, runs in $O(NL_cP_c)$. In the upper-level and lower-level data scheduling, the complexity of single reward calculation (*i.e.*, the complexity of NDCG calculation) is $O(M \log M)$, where M is the maximum length of current outgoing queue. In E_p epochs, the overall complexity of our algorithm based on GNN encoding and the PPO framework is: $O\left(E_pN\left(L_g(|E|F + |V|F^2) + L_cP_c + L_aP_a + M \log M\right)\right)$.

We have tested our algorithm on various network topologies, and the results (see Figs. 8, 9 and 10) show that it can effectively generalize across different network topologies. Remarkably, our GNN-based DRL framework can directly apply to larger topologies without requiring additional retraining. We have tested the models trained by a small-scale five-star topology (with 14 nodes) on a larger five-star one (with 25 nodes), and the results (see Fig. 10) remain reasonably good, verifying the scalability and generalization capability of our proposal.

V. Numerical Simulations

We performed simulations to compare our proposal with the state-of-the-art algorithms in the literature for routing and data scheduling of IP-DTs, including CGR [16], MARS [21], the Lyapunov optimization based approach (Lyapunov) [22], our previous scheme based on A3C (A3C) [25], and the PPO-based scheme (PPO) that uses the design of DRL models in [25] but trains them with PPO. Specifically, CGR simply schedules bundles with FIFO, while MARS considers both scenarios in [21] (MARS-1 and MARS-2). We computed each data point in the simulations by averaging the outcomes of 10 independent runs to guarantee sufficient statistical accuracy.

A. Simulation Setup

We conducted simulations on three IPNs: a four-star system with 12 nodes and two five-star ones with 14 and 25 nodes, and the Satellite Tool Kit (STK) [37] was used to generate 24-hours contact plans of them. In the four-star topology, Earth, the Moon, Mars and Jupiter are included, while the five-star topology consists of Earth, the Moon, Mars, Mercury and Jupiter. The source and destination of each bundle were randomly selected. The key parameters are listed in Table III.

[1]It is intractable to provide the theoretical analysis to formally prove the convergence of our proposal, which is a common challenge in this field.

Algorithm 4: Hierarchical DRL-based Scheduling

load parameters of DRL agents;
for *each TS* t **do**
  for *each link* $e_{v,u} \in \xi_t$ **do**
    for *each bundle* $B \in Q_{v,u}$ **do**
      get state $S^U_{v,u}$;
      get a proper action $A^U_{v,u}$ for B according to the corresponding upper-level agent;
    end
    cluster bundles in $Q_{v,u}$ with HDBSCAN method;
    put all the clusters in set Ω;
    use the median value of $\{A^U_{v,u}\}$ of all bundles in a cluster as A^U_C for them;
    for *each cluster* $C \in \Omega$ **do**
      for *each bundle* $B \in C$ **do**
        get state $S^L_{v,u}$;
        get a proper action $A^L_{v,u}$ according to the corresponding upper-level agent;
      end
      sort bundles in $Q_{v,u}$ first by A^U_C and then by $A^L_{v,u}$;
    end
  end
end

TABLE III
KEY SIMULATION PARAMETERS

Parameter	Value
TS	64 seconds
Data rate of a link	$[64, 2048]$ Kbps
TTL of bundles	6 hours
Size of bundles	[1 KB, 8 MB]

B. Training Performance

We show the training performance under the five-star system with 14 nodes to explore the scalability of our proposal. All the DRL agents in a same subnetwork (in nodes on or around a same celestial body) could share the same experience buffer to expedite training. Hence, the agents in a same subnetwork were equivalent, and we denote those with Earth, the Moon, Mars, Jupiter and Mercury as *Agents* 1-5, respectively.

1) GNN-based DRL for Routing IP-DTs: We used RELU as the activation function for each hidden layer and the Adam optimizer for training. We set the batch size as $N = 32$, the discount factor as $\gamma = 0.9$, the epsilon clip as $clip = 0.2$, and the learning rates for Actor-NN and Critic-NN as $\alpha = 0.001$ and $\beta = 0.0001$. Fig. 3 shows the training performance of the agents. During the period between $[1000, 3000]$ epochs, the rewards of all the agents exhibit significant fluctuations, indicating that the agents are actively learning and refining their strategies to achieve higher rewards, but when their training approaches $7,000$ epochs, their rewards stabilize gradually.

2) Upper-level DRL for Scheduling: We set $\alpha = 0.001$, $\beta = 0.0001$, $N = 256$, $\gamma = 0.99$, and $clip = 0.2$. Fig. 4 show that the normalized rewards of agents converge very quickly.

3) Lower-level GNN-based DRL for Scheduling: The training of these DRL agents includes two stages: the pre-training and online training. In the pre-training, we conducted offline training with the data collected from historical operations, set

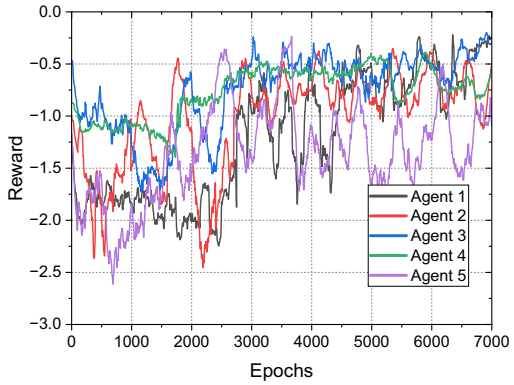


Fig. 3. Training performance of GNN-based DRL for routing IP-DTs.

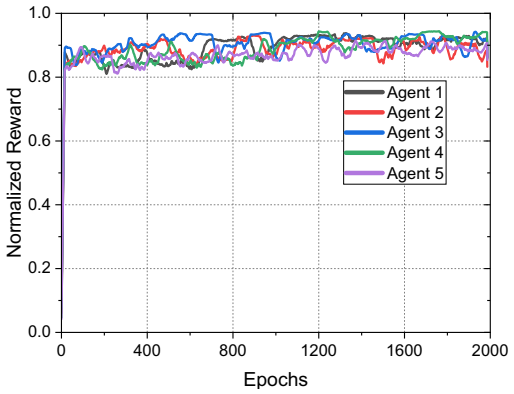


Fig. 4. Training performance of upper-level DRL for scheduling IP-DTs.

$\alpha = 0.01$, $\beta = 0.001$, $N = 256$, $\gamma = 0.99$, and $clip = 0.3$, and used LeakyRELU as the activation function. Fig. 5 indicates that after 600 epochs, the pre-training stabilizes and thus can be ended. Then, during online training, we loaded the pre-trained model in each agent and trained each agent separately, by setting $\alpha = 0.0001$, $\beta = 0.00001$, $N = 128$, $\gamma = 0.99$, and $clip = 0.3$. The performance of the online training is shown in Fig. 6. By combining Figs. 5 and 6, we can see that the pre-training significantly improves the efficiency of overall training, speeding up subsequent online training effectively.

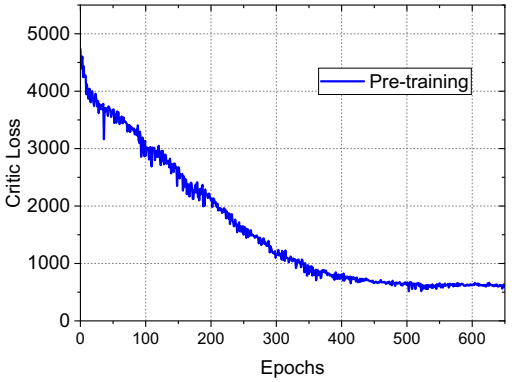


Fig. 5. Performance of pre-training of lower-level DRL for scheduling.

C. Simulations in the Four-star IPN

Fig. 7 depicts the four-star IPN system, which consists of four subnetworks for Earth, the Moon, Mars and Jupiter,

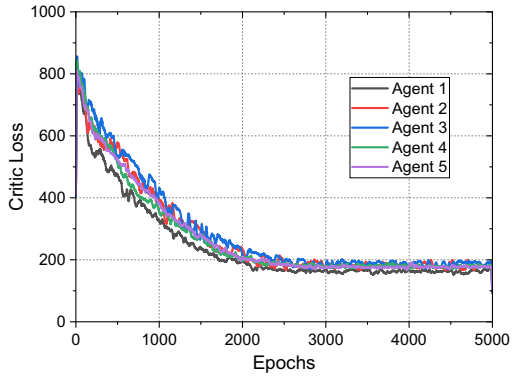


Fig. 6. Performance of online training of lower-level DRL for scheduling.

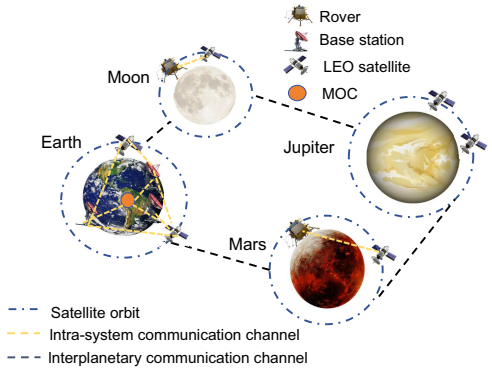
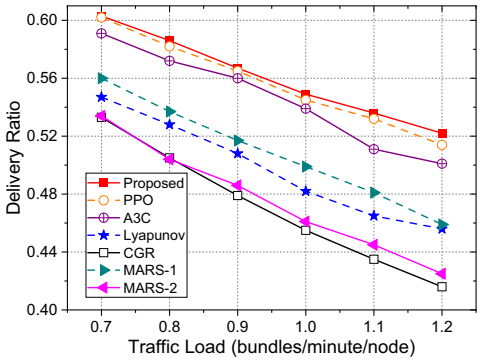
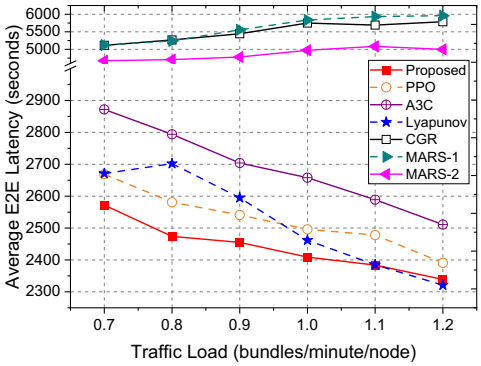


Fig. 7. Four-star IPN topology used in the simulations.



(a) Delivery ratio



(b) Average E2E latency

Fig. 8. Simulation results with four-star IPN.

respectively, where the Earth subnetwork contains an MOC, three base stations (BS'), and two satellites, each of the Moon and Mars subnetworks consists of a rover and a satellite, and the Jupiter subnetwork includes two satellites. Fig. 8 summarizes the results on delivery ratio and average E2E latency of bundles. As the traffic load increases, the delivery ratios of all the algorithms gradually decrease. It can be seen that our proposed scheme (Proposed) effectively optimizes the delivery ratio for all the traffic loads and outperforms all the benchmarks, and it also achieves the lowest average E2E latency in general, balancing the tradeoff between the two metrics the best. Note that, when the traffic load is 1.2 bundles/minute/node, the average E2E latency from Lyapunov is slightly lower than that from our proposal, but at this point its delivery ratio is much lower. Besides, the results show consistent advantages of PPO over A3C, and hereby justify our design choice of replacing A3C with PPO.

D. Simulations in the Five-star IPNs

The five-star IPNs cover Earth, the Moon, Mars, Mercury and Jupiter. In the 14-node configuration, the subnetworks of Earth, the Moon, Mars, and Jupiter use the same settings as those in the four-star IPN, and that of Mercury owns two satellites. We also expanded the 14-node configuration by adding 5, 3 and 3 satellites to the Earth, Moon and Mars subnetworks, respectively, to obtain a 25-node topology. As for the 14-node topology, the results in Fig. 9 exhibit similar trends as those in Fig. 8, further confirming the benefits of our proposal. The superior performance of our proposal lies not only in its thorough consideration of the characteristics and traffic patterns in IPNs, but also in its emphasis on leveraging the heterogeneity of different subnetworks and the connections among them. The results with the 25-node topology are presented in Fig. 10. In line with the observations drawn from the smaller-scale simulations, the proposed design exhibits good scalability and outstands all the benchmarks.

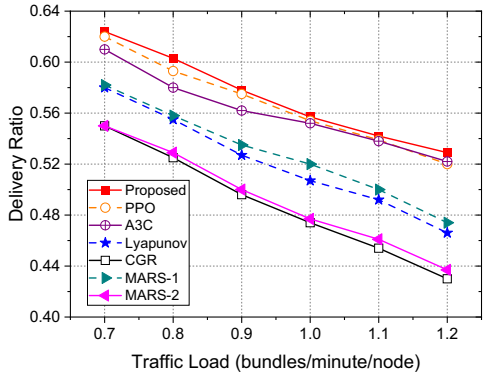
Note that, the delivery ratios from all the algorithms decrease gradually with the traffic load, which is attributed to the inherent limitations of IPNs (*e.g.*, smaller E2E bandwidth and more frequent link disruptions as the network scales), rather than the algorithms' scalability. Actually, the results in Figs. 8(a), 9(a) and 10(a) indicate enlarged performance gains achieved by our proposal over the benchmarks as IPN scales up, highlighting its robustness in handling larger and more complex network configurations.

VI. Experiments with Semi-physical IPN Emulator

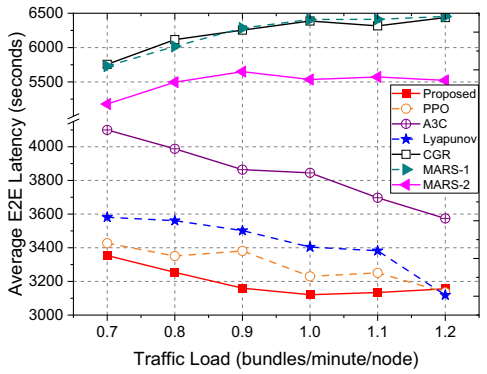
In this section, by leveraging NASA's ION software platform [28], we came up with a semi-physical IPN emulator, integrated our multi-agent DRL-based proposal in it, and conducted experiments with real data transfers between IPN nodes to further validate the performance of our proposal.

A. Implementation of IPN Emulator and Experimental Setup

NASA's ION is an open-source software platform designed to explore the challenges of DS communications in IPNs, by leveraging bundle segmentation and delay-tolerant networking (DTN) to address ultra-high latency and intermittent connections. ION supports various transport protocols, including TCP, UDP, and customized IPN-specific protocols, offering a reliable and autonomous networking framework with the

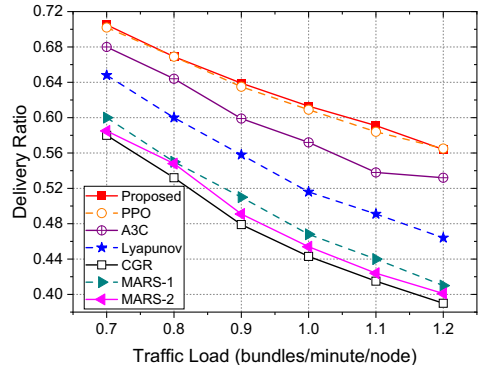


(a) Delivery ratio

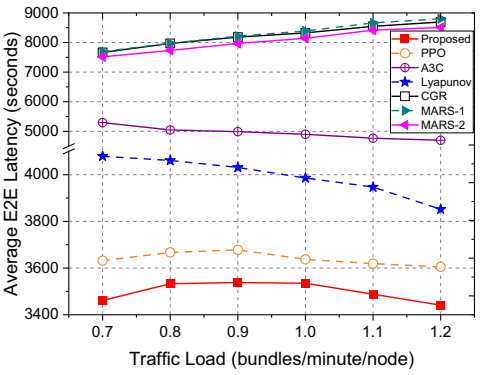


(b) Average E2E latency

Fig. 9. Simulation results with five-star 14-node IPN.



(a) Delivery ratio



(b) Average E2E latency

Fig. 10. Simulation results with five-star 25-node IPN.

flexibility and adaptability to satisfy different requirements from DS exploration missions. We implemented our multi-agent DRL-based proposal in ION. As shown in Fig. 11, we modified the IP-DT Forwarding module (marked in yellow) and added a new DRL-based Routing and Scheduling module (marked in green) in ION v4.1.2. Then, in order to verify that our proposal is lightweight enough such that excessive power, storage and computing overheads will not be introduced, we deployed the modified ION in Raspberry Pi boards, each of which contains ARM Cortex-A72 CPU and 8GB memory and runs Linux Ubuntu 20.04.5 as the operating system.

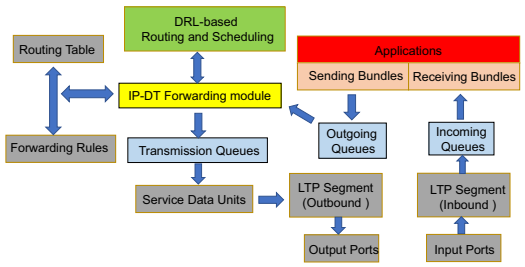


Fig. 11. Implementation of our proposal in ION.

Each Raspberry Pi board emulates the IPN nodes in a same subnetwork, and inter-subnetwork links in an IPN are emulated with the Ethernet connections between Raspberry Pi boards. Our experimental setup consists of three Raspberry Pi boards to simulate a three-star IPN that covers Earth, the Moon, and Mars with the configuration in Fig. 12. Specifically, the IPN consists of 18 nodes, where the Earth subnetwork includes three BS', an MOC, three low-orbit satellites, and one medium-orbit satellite, while the each of the Moon and Mars subnetworks has two rovers and three low-orbit satellites.

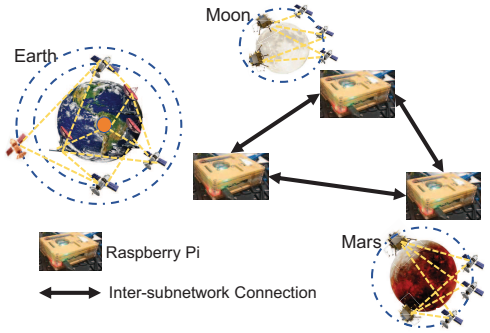
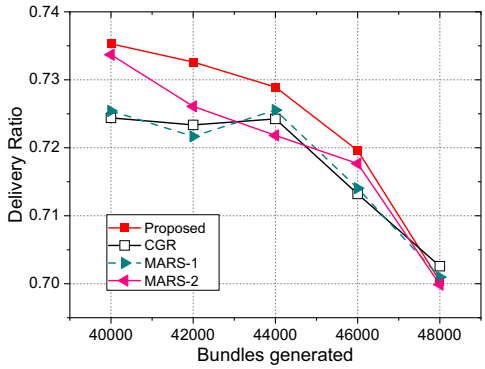


Fig. 12. Three-star topology used by the semi-physical IPN emulator.

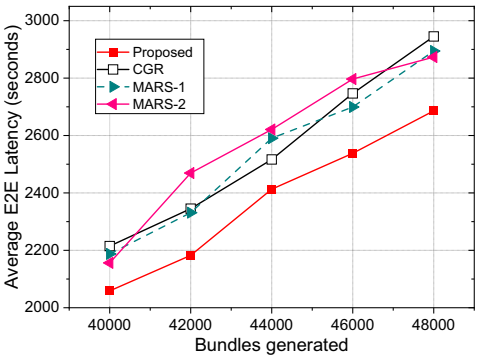
B. Experimental Results

In the experiments, we loaded trained DRL agents in the semi-physical emulator and made IPN nodes in it generate bundles with sizes within $[1\text{ KB}, 4\text{ MB}]$ and random sources and destinations to emulate real file transfers between IPN nodes. Each experiment covers 24 hours in the IPN and processes $[40000, 48000]$ bundles in total, and we obtained each data point by averaging the results from 10 independent runs. Fig. 13 shows the experimental results, which still suggest that our proposal outperforms the benchmarks. As the three-star IPN has shorter inter-subnetwork distances, the results on average E2E latency are significantly lower than the

simulation results in Section V, and for the same reason, the delivery ratios in Fig. 13 are much higher.



(a) Delivery ratio



(b) Average E2E latency

Fig. 13. Experimental results with semi-physical IPN emulator.

C. Practical Challenges

Despite the successful implementation and validation over the semi-physical emulator, deploying our multi-agent DRL system in a real IPN still faces several challenges.

First, IPN nodes like rovers and satellites have limited power budgets and processing capabilities, which presents as a major obstacle to the deployment of computing-intensive DRL agents. This challenge can be expressed by,

$$P_{\text{bgt}} \leq P_{\text{chg}} \cdot \gamma, \tag{10}$$

where P_{bgt} denotes the power budget for uninterrupted run of an IPN node, P_{chg} is the charging power, and γ is the charging duration over a unit period. Potential solutions include pre-training the agents in an IPN digital twin (*e.g.*, the semi-physical platform) offline and developing lightweight DRL models (for instance, leveraging ensemble learning).

Second, the extreme DS environment can cause unpredictable link conditions, and thereby, stochastic variations in link bandwidth and delay. We assume that disruptions occur at a rate of λ following a Poisson process, and the expected link bandwidth can be modeled as,

$$\mathbb{E}[b_{v,u,t}] = b_{v,u} \cdot \left(1 - e^{-\lambda t}\right), \tag{11}$$

where $b_{v,u}$ signifies the bandwidth of $v \rightarrow u$ in the ideal case. Further, the delay of a link can be formulated by,

$$\tau_{v,u,t} = \tau^{\text{prop}}_{v,u,t} + \tau^{\text{tran}}_{v,u,t} + \tau^{\text{queue}}_{v,u,t}. \tag{12}$$

Here, $\tau^{\text{prop}}_{v,u,t}$ denotes the propagation delay between nodes v and u, which can be calculated by,

$$\tau^{\text{prop}}_{v,u,t} = \frac{d_{v,u}}{c}, \tag{13}$$

where $d_{v,u}$ is the physical distance of the link and c is the light speed. $\tau^{\text{tran}}_{v,u,t}$ represents the transmission delay as,

$$\tau^{\text{tran}}_{v,u,t} = \frac{B_{size}}{b_{v,u,t}}. \tag{14}$$

$\tau^{\text{queue}}_{v,u,t}$ represents the queuing delay, being formulated by,

$$\tau^{\text{queue}}_{v,u,t} = \frac{\rho}{\mu(1-\rho)}, \tag{15}$$

where $\rho = \frac{\lambda_B}{\mu}$ represents the ratio of the arrival rate of bundles per node λ_B to the service rate μ.

With the modifications above, the network model of an IPN will be more realistic, ensuring more robust DRL designs that can react promptly to environment changes and learn generic (topology-invariant) knowledge. However, as the actual values of the stochastic parameters in a practical IPN are not publicly available to the best of our knowledge, we can only consider them in our future work. Last but not least, data integrity and reliable decision making are imperative for proving the practical deployment of our proposal, and thus further researches into attack proof DRL designs are necessary.

VII. Conclusion

In this work, we proposed a GNN-based multi-agent DRL approach to jointly optimize the routing and scheduling of IP-DTs in the distributed manner. Extensive simulations confirmed that our proposal can handle the routing and scheduling of IP-DTs more adaptively and balance the trade-off between delivery ratio and average E2E latency of bundles much better than the baselines. Meanwhile, by leveraging the ION developed by NASA, we built a semi-physical IPN emulator based on Raspberry Pi boards, implemented our proposal in it, and conducted experiments with real data transfers between IPN nodes, to verify that our proposal can work for practical IPNs without causing excessive overheads. Experimental results further proved the advantages of our proposal.

Acknowledgments

This work was supported by the NSFC project 62371432, and projects 2023YFB29047004 and 2022-2023-JCSS-01002.

References

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[3] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[4] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[5] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[6] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[7] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 58–66, Mar./Apr. 2017.

[8] A. Alhilal, T. Braud, and P. Hui, "The sky is NOT the limit anymore: Future architecture of the interplanetary Internet," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, pp. 22–32, Aug. 2019.

[9] C. Li *et al.*, "China's Mars exploration mission and science investigation," *Space Sci. Rev.*, vol. 217, pp. 1–24, May 2021.

[10] M. Marchese, "Interplanetary and pervasive communications," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 26, pp. 12–18, Feb. 2011.

[11] Mars. [Online]. Available: https://science.nasa.gov/mars/.

[12] Mars exploration rover telecommunications. [Online]. Available: https://descanso.jpl.nasa.gov/monograph/series13/DeepCommo_Chapter7--141030.pdf.

[13] J. Torgerson, V. Cerf, S. DeBaun, and L. Suzuki, "Space system internetworking: The foundational role of delay and disruption-tolerant networking," *IEEE J. Sel. Areas Commun.*, vol. 42, pp. 1359–1370, May 2024.

[14] R. De Gaudenzi *et al.*, "Guest editorial space communications new frontiers: From near earth to deep space," *IEEE J. Sel. Areas Commun.*, vol. 42, pp. 1023–1028, May 2024.

[15] M. Al Mamun, M. Li, and B. Pramanik, "Development of delay-tolerant networking protocols for reliable data transmission in space networks: A simulation-based approach," *IEEE Access*, vol. 12, pp. 178 642–178 658, Nov. 2024.

[16] G. Araniti *et al.*, "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Commun. Mag.*, vol. 53, pp. 38–46, Mar. 2015.

[17] E. Birrane, S. Burleigh, and N. Kasch, "Analysis of the contact graph routing algorithm: Bounding interplanetary paths," *Acta Astronaut.*, vol. 75, pp. 108–119, Jun./Jul. 2012.

[18] N. Bezirgiannidis *et al.*, "Contact graph routing enhancements for delay tolerant space communications," in *Proc. of ASMS/SPSC 2014*, pp. 17–23, Sept. 2014.

[19] S. El Alaoui and B. Ramamurthy, "EAODR: A novel routing algorithm based on the modified temporal graph network model for DTN-based interplanetary networks," *Comput. Netw.*, vol. 129, pp. 129–141, Dec. 2017.

[20] F. De Rango and M. Tropea, "DTN architecture with resource-aware rate adaptation for multiple bundle transmission in interplanetary networks," *IEEE Access*, vol. 10, pp. 47 219–47 234, Apr. 2022.

[21] S. El Alaoui and B. Ramamurthy, "MARS: A multi-attribute routing and scheduling algorithm for DTN interplanetary networks," *IEEE/ACM Trans. Netw.*, vol. 28, pp. 2065–2076, Oct. 2020.

[22] X. Tian and Z. Zhu, "On the distributed routing and data scheduling in interplanetary networks," in *Proc. of ICC 2022*, pp. 1109–1114, May 2022.

[23] ——, "On the fine-grained distributed routing and data scheduling for interplanetary data transfers," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, pp. 451–462, Aug. 2023.

[24] B. Mao, X. Zhou, J. Liu, and N. Kato, "On an intelligent hierarchical routing strategy for ultra-dense free space optical low earth orbit satellite networks," *IEEE J. Sel. Areas Commun.*, vol. 42, pp. 1219–1230, May 2024.

[25] X. Zhou, X. Tian, and Z. Zhu, "Multi-agent DRL for distributed routing and data scheduling in interplanetary networks," in *Proc. of GLOBECOM 2023*, pp. 4877–4882, Dec. 2023.

[26] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. of ICML 2016*, pp. 1928–1937, Jun. 2016.

[27] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, Aug. 2017. [Online]. Available: https://arxiv.org/abs/1707.06347.

[28] ION-DTN. [Online]. Available: https://sourceforge.net/projects/ion-dtn/.

[29] K. Scott and S. Burleigh, "Bundle protocol specification," *RFC 5050*, Nov. 2007. [Online]. Available: http://tools.ietf.org/html/rfc5050.

[30] P. Velickovic *et al.*, "Graph attention networks," in *Proc. of ICLR 2018*, Feb. 2018.

[31] L. Zhang and Z. Zhu, "Spectrum-efficient anycast in elastic optical interdatacenter networks," *Opt. Switch. Netw.*, vol. 14, pp. 250–259, Aug. 2014.

[32] W. Lu, Z. Zhu, and B. Mukherjee, "On hybrid IR and AR service provisioning in elastic optical networks," *J. Lightw. Technol.*, vol. 33, pp. 4659–4669, Nov. 2015.

[33] P. Lu and Z. Zhu, "Data-oriented task scheduling in fixed- and flexible-grid multilayer inter-DC optical networks: A comparison study," *J. Lightw. Technol.*, vol. 35, pp. 5335–5346, Dec. 2017.

[34] B. Niu *et al.*, "Visualize your IP-over-optical network in realtime: A P4-based flexible multilayer in-band network telemetry (ML-INT) system," *IEEE Access*, vol. 7, pp. 82 413–82 423, 2019.

[35] J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, pp. 1–51, Jul. 2015.

[36] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, "Learning to rank by optimizing NDCG measure," in *Proc. of NeurIPS 2009*, vol. 22, pp. 1883–1891, Dec. 2009.

[37] Satellite tool kit. [Online]. Available: http://www.agi.com/products/stk/.