Multi-Agent DRL for Distributed Routing and Data Scheduling in Interplanetary Networks

Xixuan Zhou, Xiaojian Tian and Zuqing Zhu[†]
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[†]Email: {zqzhu}@ieee.org

***Abstract*—With the fast development of deep space exploration missions, the data transfer in interplanetary networks (IPNs) is gaining increasing attention. In this work, we propose a deep reinforcement learning (DRL) based routing and data scheduling approach, which leverages a multi-agent setup for distributed operations and aims to balance the trade-off between average end-to-end (E2E) latency and delivery ratio of interplanetary data transfers (IP-DTs) well. Specifically, DRL agents based on asynchronous advantage actor-critic (A3C) are deployed on each IPN node to handle the routing and data scheduling of IP-DTs there separately. Simulation results confirm that our proposal can handle the routing and data scheduling of IP-DTs more adaptively and balance the tradeoff between the delivery ratio and average E2E latency better than the benchmarks.**

***Index Terms*—Interplanetary network (IPN), Deep reinforcement learning (DRL), Distributed routing and scheduling.**

I. Introduction

Nowadays, the Internet has been developed quickly on and around Earth with numerous new technologies [1–7], and we are also moving toward ground-breaking deep space (DS) explorations [8]. Therefore, the scope of the Internet should be further expanded to include the interplanetary networks (IPNs) [9], for enabling the communications among DS objects (*e.g.*, ground stations, satellites, landers and rovers [10]). As the example in Fig. 1 shows, IPN has several unique characteristics. First, its topology is dynamic since the movement and shields of DS objects and celestial bodies can make the connections there (*i.e.*, links in the topology) intermittent. Hence, end-to-end routing is normally infeasible for communication sessions. Second, as the orbits of DS objects and celestial bodies are usually predictable, we should plan the routing and scheduling of interplanetary data transfers (IP-DTs) accordingly, such that each contact of an intermittent link can be effectively utilized. Lastly, the links in an IPN can be several orders of magnitude longer than those in the current Internet, and thus the resulting communication latencies rule out the possibility of centralized network control and management (NC&M).

To this end, IP-DTs can only be accomplished with the store-carry-forward (SCF) scheme for delay tolerant networks (DTNs) [11]. Specifically, each source encodes outgoing data as *bundles*, each of which is an atomic unit for IP-DTs and contains sufficient information for being routed and driving the corresponding application running on its destination to make progress [12]. However, existing bundle-based techniques for IP-DTs are still in the early stage, and might have difficulty serving future IPNs, which will have larger topology scales, heavier traffic loads, and more stringent quality-of-service (QoS) demands. The national aeronautics and space administration (NASA) has proposed the contact graph routing (CGR) algorithm [13], which routes IP-DTs according to the contact plan of each link by assuming that a bundle can be sent immediately upon contact. Nevertheless, as CGR ignores the queuing delay of bundles, it becomes ineffective in a heavy-loaded IPN. Hence, CGR was enhanced to CGR-ETO [13], which considers the estimated queuing delay of bundles when choosing routing paths for them. However, it assumes that bundles are processed in the first-in-first-out (FIFO) manner, and overlooks bundle-level scheduling in queues, limiting the performance of IP-DTs with various QoS demands.

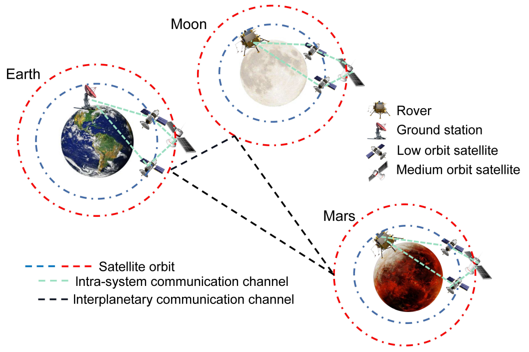


Fig. 1. An example on IPN.

Bundle-level scheduling was first tackled in [14], where the authors designed the multi-attribute routing and scheduling (MARS) algorithm that can outperform CGR and CGR-ETO in terms of bundle delivery ratio and end-to-end (E2E) IP-DT latency. Nevertheless, MARS only schedules the bundles in each queue in a greedy way but does not jointly optimize the scheduling of bundles in multiple queues on an IPN node. Note that, more than one queue can be allocated on each node for different next hops, network services, and QoS demands in future IPNs. In [15], we leveraged Lyapunov optimization to propose a distributed online algorithm for optimizing bundle scheduling in queues on each node jointly, and demonstrated that it can achieve better IP-DT performance than MARS.

Considering the complexity of routing and data scheduling in IPNs with a time-varying topology, we noticed that the approach developed in [15] still cannot solve it exactly. On the other hand, people have tried to improve CGR with machine learning (ML) in [16], and shown that ML-based Q-routing can effectively reduce the average E2E latency of IP-DTs.

Although the ML-based scheme was promising, it was still under the framework of CGR and thus did not address bundle-level scheduling. Therefore, in this work, we propose a multi-agent deep reinforcement learning (DRL) based algorithm to further improve the performance of distributed routing and data scheduling in IPNs. Specifically, each DRL agent is based on the asynchronous advantage actor-critic (A3C) framework [17, 18], and there are two types of DRL agents running on an IPN node to make intelligent decisions on how to route and schedule bundles there, respectively. Extensive simulations verify that our proposal can accomplish a better tradeoff between the average E2E latency and delivery ratio of bundles, and outperforms the existing approaches.

The rest of the paper is structured as follows. Section II elaborates on the network model and defines the distributed routing and data scheduling of IP-DTs considered in this work. We propose our multi-agent DRL based algorithm in Section III. The performance evaluations are discussed in Section IV. Finally, Section V concludes the paper.

II. Problem Formulation

A. Network Model

As the topology of each IPN is time-varying but predictable, we model it as a temporal graph $G_t(\aleph, \xi_t)$, where $\aleph$ denotes the set of existing nodes and ξ_t includes all the temporal links at time t. Note that, the IPN is a discrete-time system that operates on time-slots (TS'), each of which lasts for Δt. Then, the system time becomes $t = 0, \Delta t, 2\Delta t, \cdots$, which can be normalized as $t \in \mathcal{T} = \{0, 1, 2, \cdots\}$ for simplification [15]. Each bundle B is the atomic unit for IP-DTs, and it should be delivered to its destination before a deadline. Otherwise, its delivery is failed. Therefore, the routing and data scheduling of bundles should consider two performance metrics, *i.e.*, the delivery ratio and average E2E latency. The major notations that we use in this paper are listed in Table I.

B. Multi-Agent DRL-based Routing and Data Scheduling

Algorithm 1 shows the overall procedure of our proposed multi-agent DRL-based routing and data scheduling algorithm for IP-DTs. As the IPN is a discrete-time system, the outer for-loop of *Lines* 1-32 iterates through each TS t. *Lines* 2-3 are for the global initialization at the beginning of each TS t, where we update the time-varying topology $G_t(\aleph, \xi_t)$ and insert all the newly generated/received bundles on each node v in its local queue Q_v. Note that, similar to our work in [15], we allocate two types of queues on each node v, which are 1) a single queue Q_v that works as the first buffer of outgoing bundles, including the bundles generated locally and those using v as an intermediate node, and 2) a set of outgoing queues $\{Q_{v,u}\}$, each of which stores the bundles that will be transmitted on link $v{\rightarrow}u$ during future contacts. Next, the three for-loops that cover *Lines* 4-12, *Lines* 13-19, and *Lines* 20-30, respectively, handle the routing and data scheduling on each node and the IP-DT on each link. It can be seen that the operation on each node is actually independent, and thus *Algorithm* 1 is a distributed algorithm.

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	Source and destination of bundle B
$B_{\text{dead}}, B_{\text{size}}$	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 outgoing 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 $Q_{v,u}$
$T_{v,B}$	Queueing delay of bundle B on node v
$\left\{S^M_B, A^M_B, R^M_B\right\}$	Elements of DRL agent for routing bundle B
$\left\{S^D_{v,u}, A^D_{v,u}, R^D_{v,u}\right\}$	Elements of DRL agent for scheduling bundles in $Q_{v,u}$

The for-loop of *Lines* 4-12 goes through each node and checks each bundle $B \in Q_v$ with $B_{\text{mode}} = 0$. Here, we introduce a flag B_{mode} to determine whether DRL-based routing (*i.e.*, *Algorithm* 2) should be applied on bundle B on node v, and its default value is $B_{\text{mode}} = 0$, which means that *Algorithm* 2 is not needed. Hence, *Lines* 6-8 calculate a routing path for B with CGR if its path has not been determined yet, and *Lines* 9-10 move bundles from Q_v to $\{Q_{v,u}\}$ according to their next hops. Then, the for-loop of *Lines* 13-19 processes each bundle $B \in Q_v$ with $B_{\text{mode}} = 1$ on each node v, where the routing path of B is updated with *Algorithm* 2. Next, the data transmission on each link is handled by the for-loop of *Lines* 20-30. Specifically, if a link $e_{v,u}$ is in contact and can be used for IP-DTs, we order the bundles in $Q_{v,u}$ with DRL-based data scheduling (*i.e.*, *Algorithm* 3) and then transmit them in the scheduled order (*Lines* 21-23). Otherwise, we check each $Q_{v,u}$ on node v. If it is not empty, the bundles in it are those that were scheduled but missed their transmission opportunities. Therefore, we set $B_{\text{mode}} = 1$ to invoke rerouting for them at TS $t + 1$ (*Lines* 25-28). Finally, *Line* 31 removes the expired bundles (*i.e.*, those whose deadlines are before or at t) in all the queues in the IPN before proceeding to the next TS.

III. DRL-based Routing and Data Scheduling Algorithms

A. Background of A3C-based DRL Model

With the procedure in *Algorithm* 1, we decorrelate the routing and data scheduling of IP-DTs. Hence, each of them can be modeled as a markov decision process (MDP), and we can leverage the A3C-based DRL model [17] to design two algorithms to tackle the two subproblems separately. Specifically, each DRL model can learn the optimal strategy for routing or data scheduling of bundles automatically, by updating its parameters adaptively through interacting with the

Algorithm 1: DRL-based Routing and Data Scheduling

```
1  for each TS $t$ do
2      update $G_t(\aleph, \xi_t)$ according to current contact states;
3      insert newly generated/received bundles in $\{Q_v\}$;
4      for each node $v \in \aleph$ do
5          for each bundle $B \in Q_v$ with $B_{mode} = 0$ do
6              if $B_{path} = \emptyset$ then
7                  apply CGR to calculate $B_{\text{path}}$;
8              end
9              get next hop $u$ from $B_{\text{path}}$;
10             move $B$ from $Q_v$ to outgoing queue $Q_{v,u}$;
11         end
12     end
13     for each node $v \in \aleph$ do
14         for each bundle $B \in Q_v$ with $B_{mode} = 1$ do
15             apply Algorithm 2 to get next hop $u$ for $B$;
16             update $B_{\text{path}}$;
17             move $B$ from $Q_v$ to outgoing queue $Q_{v,u}$;
18         end
19     end
20     for each link $e_{v,u} \in \xi_t$ do
21         if $e_{v,u}$ is in contact then
22             schedule bundles in $Q_{v,u}$ with Algorithm 3;
23             transmit bundles in the scheduled order;
24         else
25             for each bundle $B \in Q_{u,v}$ do
26                 set $B_{\text{mode}} = 1$;
27                 move $B$ from $Q_{v,u}$ back to $Q_v$;
28             end
29         end
30     end
31     remove expired bundles in $\{Q_v, \{Q_{v,u}\}\}$;
32 end
```

network environment on an IPN node such that its expected long-term cumulative reward can be maximized.

Fig. 2 shows the generic framework of A3C-based DRL, which consists of a global neural network (NN) and several worker NNs. The global NN contains an actor NN and a critic NN, while each worker NN possesses the same structure as the global NN. During operation, each worker NN interacts with its own environment to gather experience data to store locally. After accumulating enough experience data, a worker NN will update its parameters accordingly and then send the new parameters to the global NN [17]. More specifically, the actor and critic NNs in it work as follows.

- **Actor NN:** The actor NN tries to keep improving the worker's decision-making strategy based on the current state to maximize the expected long-term cumulative reward. This is done by updating the actor NN's parameters in the direction of increasing the cumulative reward.
- **Critic NN:** The critic NN aims to evaluate the decision-making strategy from the actor NN with the temporal difference error, which reveals whether the decision-making strategy has been updated in the right direction.

B. DRL-based Routing for IP-DTs

As we have explained above, CGR is a classic routing algorithm for IP-DTs and it computes the routing path for sending each IP-DT with the SCF scheme based on its earliest projected delivery time, which is obtained with a rule

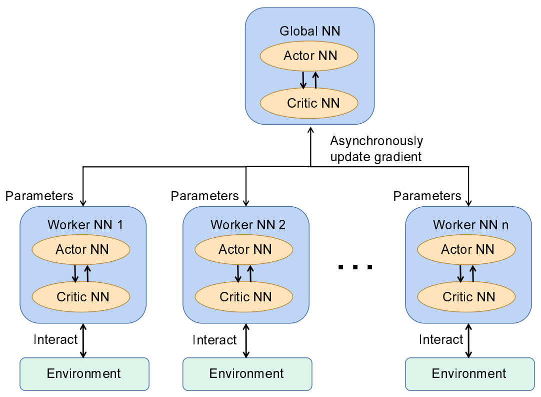


Fig. 2. Framework of A3C-based DRL.

focusing mainly on link disruptions, propagation delay and transmission latency, but ignoring the queuing delay on nodes. Nevertheless, relatively long queuing delay or even congestion can be inevitable when the traffic load is high, and thus certain bundles can miss their scheduled transmission windows. This might in turn disturb the whole SCF processes of the bundles (*i.e.*, the snowball effect). Therefore, we propose the following DRL-based routing algorithm to offer the bundles that have missed their scheduled transmission windows opportunities to update their routing paths to avoid the snowball effect.

- **Agent:** Each DRL agent runs on an IPN node to calculate routing paths for the outgoing bundles there.
- **State:** For each bundle in the local queue Q_v on node v, we define its state corresponding to each neighbor node u (*i.e.*, there is a direct link $v{\rightarrow}u$ in the IPN) to include four elements: $S_B^M = \{B_{\text{size}}, L_{v,u}, \tau_{v,u}^B, T_{v,u}^a\}$, where B_{size} and $L_{v,u}$ are defined in Table I, $\tau_{v,u}^B$ is the new projected delivery time of bundle B if it is sent out through link $v{\rightarrow}u$ and can be obtained using CGR, and $T_{v,u}^a$ is

$$T_{v,u}^a = \frac{1}{r_{v,u}} \sum_{B \in Q_{v,u}} B_{\text{size}}. \tag{1}$$

- **Action:** Each action is to use a feasible neighbor node u as the next hop of bundle B.
- **Reward:** The reward is defined to be proportional to the gap between the projected delivery time without invoking the DRL-based routing and the actual delivery time by using the next hop selected by the DRL-based routing.

The operation of the DRL agents is explained in *Algorithm* 2. *Line* 1 is for the initialization where the parameters of the actor and critic NNs of each agent are chosen randomly. Then, the for-loop of *Lines* 2-15 shows how the DRL agents for routing work in the IPN at each TS t. Specifically, the DRL agent on each node $v \in \aleph$ checks each bundle $B \in Q_v$ with $B_{\text{mode}} = 1$ and gets a next hop for it (*i.e.*, A_B^M) based on its state S_B^M (*Lines* 4-6). *Line* 7 calculates the reward R_B^M based on the actual delivery time of B. Note that, for a bundle that needs to be transferred to another planet, it might take a while for the agent to get its actual delivery time and thus *Line* 7 can be invoked at a future TS, but in order to explain the DRL agent's operation clearly, we still put it in the current iteration of TS t. Moreover, due to the intermittent connections in an

IPN, the message for reporting the actual delivery time might be lost. In this case, we will just ignore the training sample that should contain the corresponding reward. *Line* 8 stores $\left\{S_B^M, A_B^M, R_B^M\right\}$ in the agent's memory as a training sample, and *Lines* 9-12 update the parameters of the agent based on a batch of samples (*i.e.*, the online training of the agent).

Algorithm 2: DRL-based Routing for IP-DTs

```
1  initialize parameters of actor and critic NNs of agents randomly;
2  for each TS $t$ do
3      for each node $v \in \aleph$ do
4          for each bundle $B \in Q_v$ with $B_{mode} = 1$ do
5              get state $S_B^M$ and input it to DRL agent on $v$;
6              agent selects a next hop and store it in $A_B^M$;
7              agent computes reward $R_B^M$ based on the actual
               delivery time of bundle $B$;
8              push $\left\{S_B^M, A_B^M, R_B^M\right\}$ into agent's memory as a
               training sample;
9              if enough samples have been collected then
10                 agent updates its parameters based on the
                   operation principle of A3C [17];
11                 agent empties its memory for training samples;
12             end
13         end
14     end
15 end
```

C. DRL-based Data Scheduling for IP-DTs

After determining routing paths for the bundles, we still need to conduct bundle-level scheduling to ensure the performance of IP-DTs. Therefore, we design a DRL-based data scheduling algorithm, which assigns a DRL agent to manage each outgoing queue $Q_{v,u}$ on an IPN node v. Specifically, the DRL-based data scheduling is designed in consideration of the following three observations. First, we should send the bundles whose deadlines are farther (*i.e.*, those with larger time-to-live (TTL) values) earlier, because those with small TTL values can become expired even after being sent out successfully at their current nodes. Second, to reduce the average queuing delay and avoid head-of-line blocking, we should send bundles whose sizes are smaller earlier. Finally, an earlier projected delivery time for a bundle reflects a better routing path with fewer link disruptions, and thus we should send bundles with an earlier projected delivery time earlier.

Meanwhile, we leverage the following three metrics to evaluate the status of an outgoing queue.

- **Cosine Similarity (CS):** It measures the similarity between two multi-dimensional vectors $\mathbf{A}$ and $\mathbf{B}$ as
$$S_C(\mathbf{A}, \mathbf{B}) := \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||}. \tag{2}$$
- **Coefficient of Variation (CV):** It measures the dispersion of a probability distribution as
$$C_V(X) := \frac{\text{mean}(X)}{\text{var}(X)}, \tag{3}$$
where X is a random variable, mean(X) is its mean value, and var(X) is its standard deviation.

Algorithm 3: DRL-based Data Scheduling for IP-DTs

```
1  initialize parameters of actor and critic NNs of agents randomly;
2  for each TS $t$ do
3      for each link $e_{v,u} \in \xi_t$ do
4          for each bundle $B \in Q_{v,u}$ do
5              get state $S_{v,u}^D$ and input it to DRL agent of $Q_{v,u}$;
6              agent outputs an action $A_{v,u}^D$ for $B$;
7          end
8          sort bundles in $Q_{v,u}$ according to their actions;
9          agent computes reward $R_{v,u}^D$ for the scheduling result
           of each bundle $B \in Q_{v,u}$ based on $S_{v,u}^D$ and $A_{v,u}^D$;
10         push $\left\{S_{v,u}^D, A_{v,u}^D, R_{v,u}^D\right\}$ into agent's memory as a
           training sample;
11         if enough samples have been collected then
12             agent updates its parameters based on the
               operation principle of A3C [17];
13             agent empties its memory for training samples;
14         end
15     end
16 end
```

- **Normalized Discounted Cumulative Gain (NDCG):** It measures the effectiveness of a ranking system, taking into account the position of each item in the ranked list [19]. Hence, it can be used to evaluate the scheduling result for an outgoing queue, and 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 [19].

The agent, state, action and reward of a DRL agent for data scheduling are designed as follows.

- **Agent:** Each DRL agent runs on an IPN node to manage an outgoing queue $Q_{v,u}$.
- **State:** For each bundle B in an outgoing queue $Q_{v,u}$, we define its state as $S_{v,u}^D = \left\{L_{v,u}, \overline{T_{v,u}^a}, \mathbf{B}, \mathbf{C}, S_C(\mathbf{B}, \mathbf{B}^*)\right\}$, where $L_{v,u}$ is the current length of $Q_{v,u}$ in bundles, $\overline{T_{v,u}^a}$ denotes the average transmission time of bundles as
$$\overline{T_{v,u}^a} = \frac{T_{v,u}^a}{L_{v,u}}, \tag{4}$$
$\mathbf{B} = \{B_{\text{size}}, B_{\text{ttl}}, B_{\text{proj}}\}$ is the attribute vector of B, $\mathbf{C} = \{C_V(B_{\text{size}}), C_V(B_{\text{ttl}}), C_V(B_{\text{proj}})\}$ is the CV vector of bundle attributes in $Q_{v,u}$, and $S_C(\mathbf{B}, \mathbf{B}^*)$ calculates the SC between $\mathbf{B}$ and the ideal vector $\mathbf{B}^*$. Here, we have $\mathbf{B}^* = [1, 1, 1]$ after normalization.
- **Action:** The action $A_{v,u}^D$ is defined as the ranking of bundle B in $Q_{v,u}$, which is a real number within $[0, 1]$. A larger value indicates that B will be transmitted earlier.
- **Reward:** We define the reward bought by each action as
$$\begin{aligned} R_{v,u}^D =& \alpha \cdot \text{nDCG}(Q_{v,u}, B_{\text{size}}) + \beta \cdot \text{nDCG}(Q_{v,u}, B_{\text{ttl}}) \\ &+ \gamma \cdot \text{nDCG}(Q_{v,u}, B_{\text{proj}}) + \mu \cdot e^{-k \cdot T_{v,B}}, \end{aligned} \tag{5}$$
where α, β, γ are weights whose values are set according to the CVs of $\{B_{\text{size}}\}$, $\{B_{\text{ttl}}\}$, and $\{B_{\text{proj}}\}$ of bundles in $Q_{v,u}$, μ is a positive coefficient, and k is a constant to rescale the value of $T_{v,B}$ according to the position of B after scheduling. It can be seen that the first three terms in Eq. (5) use the NDCGs of $Q_{v,u}$ in terms of different attributes of bundles in it to evaluate the overall

performance of the scheduling of $Q_{v,u}$, while the last term assesses the scheduling result of bundle B.

Algorithm 3 explains our DRL-based data scheduling. This time, a DRL agent is allocated for each outgoing queue $Q_{v,u}$ in the IPN. *Lines* 4-9 show the procedure of DRL-based data scheduling for bundles in $Q_{v,u}$. Then, each agent is trained in the online way based on the principle of A3C (*Lines* 10-14).

IV. Performance Evaluation

In this section, we run simulations to compare our DRL-based approach with state-of-the-art algorithms for routing and data scheduling of IP-DTs, including CGR [13], MARS [14], and Lyapunov optimization based scheme (Lyapunov) [15].

A. Simulation Setup

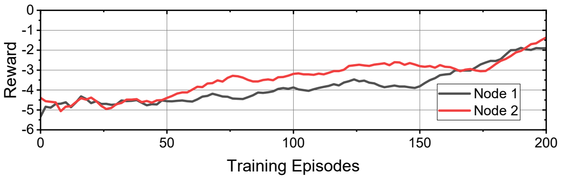
Our simulations use the satellite tool kit (STK) [20] to generate 24-hour contact plans of two Earth-Moon-Mars IPNs, which are a small-scale one with 10 nodes and a large-scale one with 18 nodes. Each bundle has an average lifetime of $7,200$ seconds and is generated on each node dynamically according to the Poisson traffic model with a random destination. Its size is uniformly distributed within $[1, 1024]$ KByte. In each IPN, the data rate of a link is set within $[32, 128]$ Kbps, according to the settings of practical DS communications [9, 10]. The time granularity of routing and data scheduling of IP-DTs is $\Delta t = 1$ second (*i.e.*, t increments by seconds). The parameters of our DRL models are listed in Table II. To ensure sufficient statistical accuracy, we average the results of 10 independent runs to get each data point in the simulations.

TABLE II
PARAMETERS OF DRL MODELS

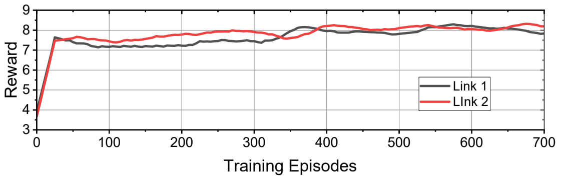
Parameters	*Algorithm* 2	*Algorithm* 3
Number of Workers	16	
Optimizer	Adam	
Discount Factor	0.99	
Learning Rate of Actor NNs	0.01	$[0.0001, 0.1]$
Learning Rate of Critic NNs	0.001	$[0.0001, 0.01]$
Batch Size of Training Samples	128	512
Layers of Actor NN/Critic NN	2/6	3/6

B. Small-Scale Simulations

The small-scale IPN consists of three subsystems on/around Earth, Moon and Mars, respectively, each of which contains a rover and a relay satellite, along with three base stations (BS') and a mission operation center (MOC) on Earth. Fig. 3 shows the training performance of our multi-agent DRL models. As there are many DRL agents, we just select those for two nodes and two outgoing queues and plot their reward changes in Figs. 3(a) and 3(b), respectively. Here, *Nodes* 1 and 2 are the relay satellite around Earth and Mars, respectively, and *Link 1* denotes the link between *Node 1* and a BS, and *Link 2* is the link between *Nodes* 1 and 2. The rewards increase and converge quickly in online training, especially for the DRL agents for data scheduling. We also confirm that other DRL agents have similar training performance in the simulations.



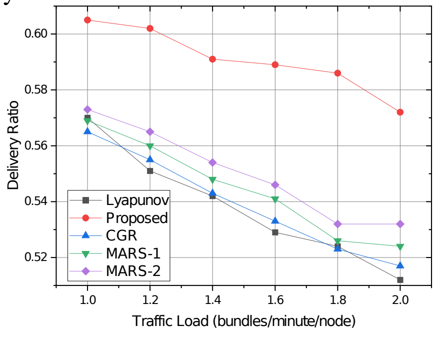
(a) Reward of DRL-based routing



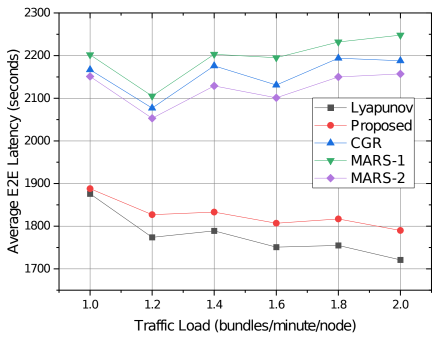
(b) Reward of DRL-based data scheduling

Fig. 3. Training performance with the small-scale IPN.

Fig. 4 shows the results on delivery ratio and average E2E latency of bundles. In Fig. 4(a), we observe that our proposed DRL-based approach achieves significantly higher delivery ratio than all the benchmarks. The delivery ratios from CGR and Lyapunov are similar, and they are both smaller than the two MARS-based algorithms. The results in Fig. 4(b) indicate that the average E2E latency from our proposal is much shorter than those from CGR and MARS-based algorithms, and it is only slightly longer than that from Lyapunov. Hence, Fig. 4 confirms that our proposed DRL-based approach can handle the routing and data scheduling of IP-DTs more adaptively and balance the tradeoff between the delivery ratio and average E2E latency of bundles better than the benchmarks.



(a) Delivery ratio



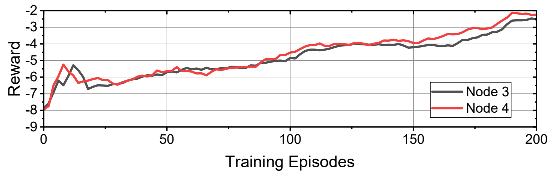
(b) Average E2E latency

Fig. 4. Simulation results with the small-scale IPN.

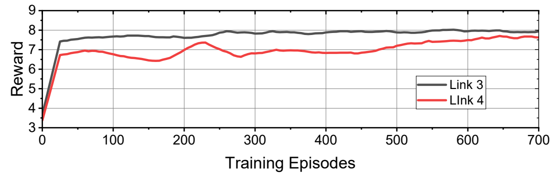
C. Large-Scale Simulations

The large-scale IPN still consists of three subsystems. The subsystem on/around Earth contains three base stations, a

MOC, three low orbit satellites, and a medium orbit satellite, while each of those on/around Moon and Mars includes two rovers, two low orbit satellites, and a medium orbit satellite. Fig. 5 shows the training performance of our multi-agent DRL models. Here, *Nodes* 3 and 4 denote a low orbit satellite around Earth and the medium orbit satellite around Mars, respectively. *Link* 3 is the link between *Nodes* 3 and 4, and *Link* 4 is the link between the medium orbit satellite around Earth and *Node 4*. We can see that the rewards of the DRL agents converge similarly as those in Fig. 3, verifying that our multi-agent DRL can adapt to the dynamic IPN quickly in online training.



(a) Reward of DRL-based routing



(b) Reward of DRL-based data scheduling

Fig. 5. Training performance with the large-scale IPN.

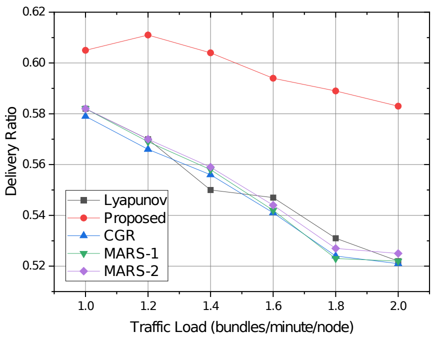
The results of large-scale simulations are shown in Fig. 6, and similar trends as those in Fig. 4 can be observed. This verifies that our proposed DRL-based approach can balance the tradeoff between the delivery ratio and average E2E latency of bundles better than the benchmarks, regardless of the topology of an IPN. Moreover, it is interesting to notice that the gaps between our proposal and all the benchmarks actually increase with the traffic load. This is because the large-scale IPN has better connectivity and thus provides more options for our DRL-based approach to handle IP-DTs adaptively.

V. Conclusion

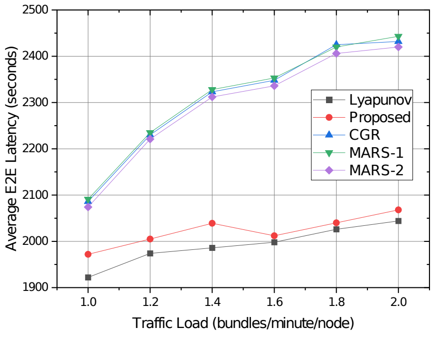
In this work, we proposed a multi-agent DRL-based approach for the routing and data scheduling of IP-DTs, which optimizes the routing and data scheduling of bundles on each node separately. Simulations confirmed that our proposal can handle the routing and data scheduling of IP-DTs more adaptively and balance the tradeoff between the delivery ratio and average E2E latency of bundles better than the benchmarks.

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



(a) Delivery ratio



(b) Average E2E latency

Fig. 6. Simulation results with the large-scale IPN.

[5] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.
[6] S. Li, D. Hu, W. Fang, and Z. Zhu, "Source routing with protocol-oblivious forwarding (POF) to enable efficient e-health data transfers," in *Proc. of ICC 2016*, pp. 1–6, Jun. 2016.
[7] C. Chen *et al.*, "Demonstrations of efficient online spectrum defragmentation in software-defined elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 4701–4711, Dec. 2014.
[8] C. Li *et al.*, "China's Mars exploration mission and science investigation," *Space Sci. Rev.*, vol. 217, pp. 1–24, May 2021.
[9] 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.
[10] M. Marchese, "Interplanetary and pervasive communications," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 26, pp. 12–18, Feb. 2011.
[11] V. Cerf *et al.*, "Delay-tolerant networking architecture," *RFC 4838*, Apr. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4838.
[12] K. Scott and S. Burleigh, "Bundle protocol specification," *RFC 5050*, Nov. 2007. [Online]. Available: http://tools.ietf.org/html/rfc5050.
[13] 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.
[14] 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.
[15] 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.
[16] R. Dudukovich, A. Hylton, and C. Papachristou, "A machine learning concept for DTN routing," in *Proc. of WiSEE 2017*, pp. 110–115, Oct. 2017.
[17] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. of ICML 2016*, pp. 1928–1937, Jun. 2016.
[18] X. Tian, B. Li, R. Gu, and Z. Zhu, "Reconfiguring multicast sessions in elastic optical networks adaptively with graph-aware deep reinforcement learning," *J. Opt. Commun. Netw*, vol. 13, pp. 253–265, Jul. 2021.
[19] 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.
[20] Satellite tool kit. [Online]. Available: http://www.agi.com/products/stk/.