Deep Reinforcement Learning Aided Packet-Routing For Aeronautical Ad-Hoc Networks Formed by Passenger Planes

Dong Liu, Jingjing Cui, Jiankang Zhang, Chenyang Yang, and Lajos Hanzo

***Abstract*—Data packet routing in aeronautical ad-hoc networks (AANETs) is challenging due to their high-dynamic topology. In this paper, we invoke deep reinforcement learning for routing in AANETs aiming at minimizing the end-to-end (E2E) delay. Specifically, a deep Q-network (DQN) is conceived for capturing the relationship between the optimal routing decision and the local geographic information observed by the forwarding node. The DQN is trained in an offline manner based on historical flight data and then stored by each airplane for assisting their routing decisions during flight. To boost the learning efficiency and the online adaptability of the proposed DQN-routing, we further exploit the knowledge concerning the system's dynamics by using a deep value network (DVN) conceived with a feedback mechanism. Our simulation results show that both DQN-routing and DVN-routing achieve lower E2E delay than the benchmark protocol, and DVN-routing performs similarly to the optimal routing that relies on perfect global information.**

***Index Terms*—AANET, routing, deep reinforcement learning**

I. Introduction

Next-generation wireless systems are expected to support global communications, anywhere and anytime [1]. Current in-flight Internet access supported by geostationary satellites or direct air-to-ground (A2G) communications typically exhibit either high latency or limited coverage. Aeronautical ad-hoc networks (AANETs) are potentially capable of extending the coverage of A2G networks by relying on commercial passenger airplanes to act as relays for forming a self-configured wireless network via multihop air-to-air (A2A) communication links [2].

Due to the high velocity of aircraft and the distributed nature of ad-hoc networking, one of the fundamental challenges in AANETs is to design an efficient routing protocol for constructing an appropriate path for data transmission at any given time. Traditional topology-based ad-hoc routing protocols [3] usually require each node to locally store a routing table specifying the next hop. The routing table, however, has to be refreshed whenever the network topology changes during a communication session, hence imposing substantial signaling overhead and latency in AANETs. Although research efforts have been invested for improving the stability of routing in AANETs [4, 5], they have a limited ability to update the routing tables for prompt adaption in high-dynamic scenarios.

D. Liu, J. Cui, and L. Hanzo are with the School of Electronics and Computer Science, the University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: d.liu@soton.ac.uk; jingj.cui@soton.ac.uk; lh@ecs.soton.ac.uk).

J. Zhang is with the Department of Computing and Informatics, Bournemouth University, Bournemouth BH12 5BB, U.K. (e-mail: jzhang3@bournemouth.ac.uk).

C. Yang is with the School of Electronics and Information Engineering, Beihang University, Beijing 100191, China (e-mail: cyyang@buaa.edu.cn).

This work was supported in part by the Engineering and Physical Sciences Research Council Projects under Grant EP/N004558/1, Grant EP/P034284/1, Grant EP/P034284/1, and Grant EP/P003990/1 (COALESCE), in part by the Royal Society's Global Challenges Research Fund Grant, and in part by the European Research Council's Advanced Fellow Grant QuantCom (Grant No. 789028).

The code for reproducing the results of this paper is available at https://github.com/Fluidy/tvtl2020.

By contrast, another family of ad-hoc routing protocols, namely position-based (or geographic) routing [6], only requires the position information of the single-hop neighbors and of the destination for determining the next hop. Since it does not have to maintain routing tables, geographic routing finds new routes almost instantly, when the topology changes. Because the position information required can be readily obtained by each airplane using the automatic dependent surveillance-broadcast system on board, geographic routing is more appealing in AANETs. Greedy perimeter stateless routing (GPSR) [7] was one of the most popular geographic routing protocols, which has also inspired various extensions [8, 9] in AANETs. The core idea of greedy routing is to forward the packet to the specific neighbor that is geographically closest to the destination. In [8], greedy routing was improved for avoiding congestion by considering the queue status of next hop. In [9], the mobility information was further taken into consideration for choosing a more stable next hop. However, the performance of greedy routing [7–9] suffers when no neighbor is closer to the destination than the forwarding node (such a situation is term as the *communication void*).

To elaborate, the limitation of greedy-based routing arises from the fact that the nodes are unaware of the entire network topology. Therefore, our ambitious goal is to enable the forwarding node to infer the global topology from its local observation for bypassing the communication void more efficiently. Although the topology of AANETs revolves dynamically, it exhibits certain patterns, since the flight path and takeoff time are preplanned and remain fairly similar on the same day of different weeks. This suggests that the local geographic information may be strongly correlated with the whole topology, and such correlation may be learned from historical flight data. In this context, recent advances in deep reinforcement learning (DRL) [10] have demonstrated the powerful capability of deep neural networks (DNNs) for learning a direct mapping from the observation gleaned to the desired action to be taken.

Against this background, we invoke DRL for routing in AANETs aiming at minimizing the end-to-end (E2E) delay. Our major contributions can be summarized as follows:

1) We propose a DRL-based routing algorithm using deep Q-network (DQN) [10], for directly mapping local geographic information to optimal routing decisions. Distinguished from routing algorithms based on tabular-based reinforcement learning (RL), such as Q-routing [11] and its variants [12], which requires frequent information exchange for updating the Q-table online whenever the topology changes, our proposed DQN can be trained offline based on historical flight data to "embed" the global network topology. During its flight, the forwarding node can infer the information required for deciding the next-hop by inputting its local observation into the DQN, without requiring any online update.

2) To boost the learning efficiency, we further design another routing algorithm based on deep value-network (DVN) by exploiting the knowledge concerning the system's dynamics. Moreover, we introduce a feedback mechanism so that the forwarding node is able to plan one step ahead for enhancing the online adaptability.
3) Our simulation results show that both DQN-routing and DVN-routing achieve lower E2E delay than GPSR. Furthermore, the performance gap between DVN-routing and the optimal routing relying on perfect global information is marginal.

II. System Model

Consider an AANET formed by passenger airplanes. Two nodes can establish direct communication link when they are above the radio horizon. The delay of the direct link from node i to node j can be expressed as

$$D_{\text{link}}(i,j) = \frac{d(i,j)}{c} + \frac{S}{R(i,j)}, \tag{1}$$

where the first and second terms are the propagation delay and transmission delay, respectively, $d(i,j)$ denotes the distance between nodes i and j, c is the speed of light, S is the packet size, and $R(i,j)$ denotes the data rate of the link $i \to j$. We consider the decode-and-forward relaying protocol and let $D_{\text{que}}(i)$ denote the queuing delay at node i.

Our goal is to find the optimal route $\mathcal{P} = (i_1, \cdots, i_T)$ minimizing the E2E packet delay between an source node i_{s} and destination i_{d}, which can be formulated as

$$\min_{\mathcal{P}} \quad \sum\nolimits_{t=1}^{T-1} \left[D_{\text{que}}(i_t) + D_{\text{link}}(i_t, i_{t+1})\right] \tag{2a}$$

$$s.t. \quad I(i_t, i_{t+1}) = 1, \ \forall t = 1, \cdots, T-1, \tag{2b}$$

where $I(i_t, i_{t+1}) = 1$ if nodes i_t and i_{t+1} are above the radio horizon and $I(i_t, i_{t+1}) = 0$ otherwise, $i_1 = i_{\text{s}}$ and $i_T = i_{\text{d}}$.

Problem (2) can be solved by classic shortest path search algorithms, which requires the global information regarding the queuing delay of each node and the link delay between every two nodes. However, the node positions change rapidly due to the high velocity of airplanes. Consequently, this may impose substantial signaling overhead by keeping the required information up-to-date for implementing the algorithm.

In the following, we assume that each node is only aware of its own position, the positions of the nodes within its direct communication range (i.e., its *neighbors*) as well as the destination, and invoke DRL for finding the optimal route in an distributed manner.

III. DRL for Geographic Routing

In this section, we first recast the routing problem (2) into the RL framework by designing the key elements of RL, and propose our DRL-based routing policies.

A. RL framework

In a RL problem, an agent learns from its interactions with the environment for achieving a desired goal [13]. At each *time step* t, the agent observes the *state* $\mathbf{s}_t$ of the environment and on that basis executes an *action* a_t. Then, the agent receives a *reward* r_{t+1} from the environment and transits into a new state $\mathbf{s}_{t+1}$. The goal of the agent is to learn a mapping from $\mathbf{s}_t$ to a_t (i.e., a policy π) for minimizing[1] an expected *return* $\mathbb{E}\big[\sum_{t=1}^{T-1} \gamma^{t-1} r_{t+1}\big]$, which reflects the accumulated reward received by the agent during an *episode* with T time steps.

In the routing problem (2), we specify that a time step starts when a node has received a packet and ends when the packet has been transmitted to the next node. Then, the whole episode begins when the packet is generated by the source and ends when the packet has been received by the destination or fail to reach the destination within $t_{\max}$ hops. Different from existing RL-based routing algorithms that train a distinct agent for each individual node, the agent in our framework moves along with the packet, and all the nodes share the agent's parameters, which improves the learning efficiency and the scalability.

Let i_t denote the node where the packet is located at the beginning of time step t, and let $\mathbf{x}(i) = (\mathsf{longitude}_i\ [\text{E}^\circ], \mathsf{latitude}_i\ [\text{N}^\circ], \mathsf{altitude}_i\ [\text{km}])$ denote the position of node i. Then, the current position of the packet and of the destination can be denoted as $\mathbf{x}(i_t)$ and $\mathbf{x}(i_{\text{d}})$, respectively. Let $\mathcal{N}_{i_t} \triangleq \{j | I(i_t, j) = 1\}$ denote the neighbors of node i_t. To limit the dimension of action exploration, the neighbors are ranked by their distances to the destination in ascending order, and the next hop is selected only from neighbors ranked among the top K. They are termed as the *candidates* and denoted by $\mathcal{C}_{i_t} \triangleq \{i_t^1, \cdots, i_t^K\}$, where i_t^k represents the kth-ranking neighbor (candidate) of node i_t.

Action: In time step t, the forwarding node i_t should determine which candidate is selected as the next hop. Therefore, the action a_t can be represented by the ranking of the candidate to be selected. Then, the next hop is node $i_t^{a_t}$.

State: Since our goal is to learn a routing policy that only depends on local geographical information. The state is designed to include the positions of the source and the destination, as well as on the positions of the candidates, i.e.,

$$\mathbf{s}_t = \big[\mathbf{x}(i_t), \mathbf{x}(i_t^1), \cdots, \mathbf{x}(i_t^K), \mathbf{x}(i_{\text{d}})\big] \triangleq \mathbf{s}(i_t), \tag{3}$$

where we introduce the notation $\mathbf{s}(i_t)$ to emphasize that $\mathbf{s}_t$ is the local geographic information observed by node i_t and we will use $\mathbf{s}_t$ and $\mathbf{s}(i_t)$ interchangeably in the following.

Reward: The reward function can be naturally designed as the delay experienced within time step t,

$$r_{t+1} = D_{\text{que}}(i_t) + D_{\text{link}}(i_t, i_{t+1}). \tag{4}$$

In this way, minimizing the return is equivalent to minimizing the average E2E delay.

The *action-value function* is defined as [13]

$$Q_\pi(\mathbf{s}_t, a_t) \triangleq \mathbb{E}\Big[\sum\nolimits_{l=t}^{T-1} \gamma^{l-t} r_{l+1} \ \Big|\ \mathbf{s}_t, a_t, \pi\Big]. \tag{5}$$

For the routing problem considered, we set $\gamma = 1$ and hence $Q_\pi(\mathbf{s}_t, a_t)$ represents the delay between the forwarding node and the destination by selecting node $i_t^{a_t}$ as the next hop and thereafter forwarding the packet according to policy π. Then, the *optimal action-value function* is defined as $Q_*(\mathbf{s}_t, a) \triangleq \min_\pi Q_\pi(\mathbf{s}_t, a_t)$, from which the optimal policy

[1] In contrast to a standard RL problem defined to maximize the return, we consider minimizing the return because we aim at minimizing the E2E delay.

can be readily obtained as $\pi_*(\mathbf{s}_t) = \arg\min_a Q_*(\mathbf{s}_t, a)$. In this sense, $Q_*(\cdot)$ contains all the information required for determining the optimal next hop, or in other words, it embeds the global network topology. Therefore, the agent's goal can be accomplished by learning $Q_*(\cdot)$.

The Bellman equation for $Q_*(\cdot)$ can be expressed as [13]

$$Q_*(\mathbf{s}_t, a_t) = \mathbb{E}\big[r_t + \min_a Q_*(\mathbf{s}_{t+1}, a) \mid \mathbf{s}_t, a_t, \pi_*\big], \quad (6)$$

based on which various RL algorithms, such as Q-learning [13], have been developed to learn the optimal action value function. However, Q-learning is faced with the curse of dimensionality due to the continuous nature of the state $\mathbf{s}_t$. Thus, we resort to DRL, specifically DQN, for learning the optimal action value function.

B. DQN-Routing

We employ a DNN $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_Q)$ shared by all the nodes to learn the optimal action-value function $Q_*(\mathbf{s}_t, a_t)$. The DQN parameter $\boldsymbol{\theta}_Q$ is trained offline using the historical flight trajectories. Specifically, we create a large set of snapshots containing the position of each flight at each timestamp.

During the offline training phase, the transmission delay and propagation delay can be calculated based on the flight positions. As for the queuing delay, since we aim to train the DQN for embedding the historical topology information, which is independent from the packet traffic, we assume that the queuing delay is identical and constant among all the nodes during training. In this way, the total queuing delay is actually determined by the number of hops in the route.

Let each node forward its received packet in a ε-greedy manner, i.e., with probability ε randomly selecting an action for exploration and with probability $1-\varepsilon$ selecting action $a_t = \arg\min_a Q(\mathbf{s}_t, a; \boldsymbol{\theta}_Q)$ for exploitation. Everytime a packet is forwarded to the next hop, the experience vector $\mathbf{e}_t = [\mathbf{s}_t, a_t, r_{t+1}, \mathbf{s}_{t+1}]$ is recorded in a replay memory $\mathcal{D}$ and we randomly sample a batch of experiences $\mathcal{B}$ from $\mathcal{D}$ for updating the parameter $\boldsymbol{\theta}_Q$ (i.e., the experience replay [10]). Based on the Bellman equation (6), $\boldsymbol{\theta}_Q$ is updated by minimizing the loss function $\mathbb{E}[(y_t - Q(\mathbf{s}_t, a_t; \theta_Q))^2]$ using stochastic gradient descent $\boldsymbol{\theta}_Q \leftarrow \boldsymbol{\theta}_Q - \frac{\delta}{|\mathcal{B}|}\nabla_{\boldsymbol{\theta}_Q} \sum_{\mathbf{e}_l \in \mathcal{B}} [y_l - Q(\mathbf{s}_l, a_l; \boldsymbol{\theta}_Q)]^2$, where δ is the learning rate, $y_l = r_{l+1}$ if the episode ends on state $\mathbf{s}_{l+1}$, and $y_l = r_{l+1} + Q'(\mathbf{s}_{l+1}, \arg\min_a Q(\mathbf{s}_{l+1}, a; \boldsymbol{\theta}_Q); \boldsymbol{\theta}'_Q)$ otherwise. Furthermore, $Q'(\cdot; \boldsymbol{\theta}'_Q)$ represents the target network, which has the same structure as the DQN $Q(\cdot; \boldsymbol{\theta}_Q)$ and is updated by $\boldsymbol{\theta}'_Q \leftarrow \tau\boldsymbol{\theta}_Q + (1-\tau)\boldsymbol{\theta}'_Q$ with very small value of τ to reduce the correlations between the action value $Q(\mathbf{s}_l, a_l; \boldsymbol{\theta}_Q)$ and the target values y_l [10].

After the training converges, the DQN can be copied to each airplane in support of online routing decisions. During its flight, each airplane forwards its received packet according to the DQN based on the state it observes. Specifically, node i_t observe its state $\mathbf{s}(i_t)$ and then evaluates

$$a_t^* = \arg\min_{a \in \mathcal{A}_t} \left[Q(\mathbf{s}(i_t), a; \boldsymbol{\theta}_Q)\right], \quad (7)$$

where $\mathcal{A}_t \triangleq \{k | 1 \le k \le K, i_t^k \ne i_1, \cdots, i_{t-1}\}$ specifics that the next hop cannot be chosen from the previously selected nodes to avoid loops in the routes. Then, node i_t forwards its received packet to node $i^{a_t^*}$.

The above implementation of DQN represents a generic approach to solving completely model-free RL problems. However, in the considered routing problem, the system's dynamics can be partially known, which can be exploited for faster learning and better online adaptability. Moreover, the training of DQN treats the queuing delay as an identical constant, while in reality the queuing delay varies due to different packet arrival rate. In the following, we develop a specialized DRL algorithm for learning the optimal routing policy more efficiently and introduce a feedback mechanism for taking the real-time queuing delay into consideration.

C. DVN-Routing With Feedback

In this subsection, we first specify the knowledge concerning the system's dynamics, which is then exploited for boosting the learning efficiency. Then, based on the feedback received from the next-hop candidates, the forwarding node is capable of planning one step ahead before forwarding the packet, which improves the online adaptability of the policy.

1) Exploiting the System's Dynamics: For the routing problem of AANETs, given the current state $\mathbf{s}_t$ and an arbitrary action a_t, the next state can be predicted before the forwarding node sends the packet, because the movement of nodes can be neglected within a single time step.[2] Specifically, the next state $\mathbf{s}_{t+1}$ is actually the state observed by node $i_t^{a_t}$ in time step t, which yields

$$\mathbf{s}_{t+1} = \mathbf{s}(i_t^{a_t}) = \big[\mathbf{x}(i_t^{a_t}), \mathbf{x}(i_t^{a_t,1}), \cdots, \mathbf{x}(i_t^{a_t,K}), \mathbf{x}(i_{\mathrm{d}})\big], \quad (8)$$

where $i_t^{a_t,k}$ denotes the kth candidate of node $i_t^{a_t}$.

As for the reward, before i_t forwards a packet, the link delay $D_{\mathrm{link}}(i_t, i_t^{a_t})$ can actually be computed in advance according to (1) at the next hop $i_t^{a_t}$ and the queuing delay $D_{\mathrm{que}}(i_t^a)$ can also be measured by i_t^a based on its queuing status [8].

To exploit the above knowledge regarding the state transition and the reward, we introduce the *intermediate-state-value function* of a routing policy π, defined by

$$V_\pi(\mathbf{s}_t) \triangleq \mathbb{E}\Big[D_{\mathrm{link}}(i_t, i_t^{a_t}) + \sum\nolimits_{l=t+1}^{T} r_{l+1} \;\Big|\; \mathbf{s}_t, \pi\Big], \quad (9)$$

which captures the expected delay commencing from the instant when the packet has experienced its queuing delay at node i_t until it reaches its final destination, by forwarding according to π. Correspondingly, the *optimal intermediate-state-value function* is defined as $V_*(\mathbf{s}_t) \triangleq \min_\pi V_\pi(\mathbf{s}_t)$.

Bearing in mind the definitions of $V_*(\cdot)$ and $Q_*(\cdot)$ as well as r_{t+1}, we can write $V_*(\cdot)$ in terms of $Q_*(\cdot)$ as

$$V_*(\mathbf{s}_t) + \mathbb{E}\left[D_{\mathrm{que}}(i_t)\right] = \min_a Q_*(\mathbf{s}_t, a), \quad (10)$$

and write $Q_*(\cdot)$ in terms of $V_*(\cdot)$ as

$$Q_*(\mathbf{s}_t, a) = \mathbb{E}\left[r_{t+1} + D_{\mathrm{que}}(i_t^a)\right] + V_*(\mathbf{s}_{t+1}). \quad (11)$$

Observe from (11) that the value of $Q_*(\cdot)$ can be obtained by learning $V_*(\cdot)$ instead. Then, by substituting (11) into (10)

[2]For a flight cruise speed of 900 km/h, the position shift within a typical time step of 10 ms is only 2.5 m, which is much smaller than the minimum distance allowed between airplanes and hence can be safely neglected.

and considering $\mathbf{s}_{t+1} = \mathbf{s}(i_t^a)$, we can obtain the Bellman equation for $V_*(\cdot)$ as

$$V_*(\mathbf{s}_t) = \min_a \left\{ \mathbb{E}\left[r(i_t, i_t^a)\right] + V_*\big(\mathbf{s}(i_t^a)\big) \right\}, \tag{12}$$

where $r(i_t, i_t^a) \triangleq D_{\text{link}}(i_t, i_t^a) + D_{\text{que}}(i_t^a)$.

2) Offline Training: Similarly to DQN-routing, we invoke a DNN $V(\mathbf{s}_t; \boldsymbol{\theta}_V)$ to learn $V_*(\mathbf{s}_t)$, termed as the DVN. In contrast to DQN, the scale of DVN can be much smaller, because it does not depend on the action, and hence has less parameters to train.

During the offline training phase, again, we use the historical flight data and assume constant and identical queuing delay. Let each node forward its received packet in a ε-greedy manner. According to (12), the action for exploitation is determined by

$$a_t = \arg\min_a \left[r(i_t, i_t^a) + V\big(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V\big) \right]. \tag{13}$$

Then, the experience vector composed by

$$\tilde{\mathbf{e}}_t = [\mathbf{s}_t, \mathbf{s}(i_t^1), \cdots, \mathbf{s}(i_t^K), r(i_t, i_t^1), \cdots, r(i_t, i_t^K)] \tag{14}$$

is recorded in the replay memory $\mathcal{D}$ and we randomly sample a batch of experiences $\mathcal{B}$ from $\mathcal{D}$. Based on the Bellman equation (12), θ_V is updated by minimizing the loss function $\mathbb{E}[y_t - V(\mathbf{s}_t; \boldsymbol{\theta}_V)]$ via stochastic gradient descent as

$$\boldsymbol{\theta}_V \leftarrow \boldsymbol{\theta}_V + \frac{\delta}{|\mathcal{B}|} \sum\nolimits_{\tilde{\mathbf{e}}_l \in \mathcal{B}} \left[y_{l+1} - V(\mathbf{s}_l; \boldsymbol{\theta}_V) \right]^2, \tag{15}$$

where $y_l = r(i_l, i_l^{a_*})$ if $i_l^{a_*} = i_{\text{d}}$, $y_l = r(i_l, i_l^{a_*}) + V'\big(\mathbf{s}(i_l^{a_*}); \boldsymbol{\theta}'_V\big)$ otherwise, $a_* = \arg\min_a \left[r(i_l, i_t^a) + V\big(\mathbf{s}(i_l^a); \boldsymbol{\theta}_V\big) \right]$, and finally $V'(\cdot; \boldsymbol{\theta}'_V)$ is the target network updated by $\boldsymbol{\theta}'_V \leftarrow \tau\boldsymbol{\theta}_V + (1-\tau)\boldsymbol{\theta}'_V$.

3) Online Decision: Once sufficiently well trained, the DVN is copied to each airplane for online routing decision. Since the information required for determining the action in (13), i.e., $r(i_t, i_t^a)$ and $V\big(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V\big)$ for $a = 1, \cdots, K$, are only available at the next-hop candidates, we introduce a feedback mechanism for enabling the forwarding node i_t to obtain these information. Specifically, each candidate i_t^a estimates $D_{\text{link}}(i_t, i_t^a)$ and $D_{\text{que}}(i_t^a)$, observes its state $\mathbf{s}(i_t^a)$ and computes $V\big(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V\big)$, and then sends $r(i_t, i_t^a) + V\big(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V\big)$ to the forwarding node i_t, as shown in Fig. 1. Finally, the forwarding node i_t selects the action

$$a_t^* = \arg\min_{a \in \mathcal{A}_t} \left[r(i_t, i_t^a) + V\big(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V\big) \right], \tag{16}$$

where $\mathcal{A}_t$ is used for avoiding loops in the routes.

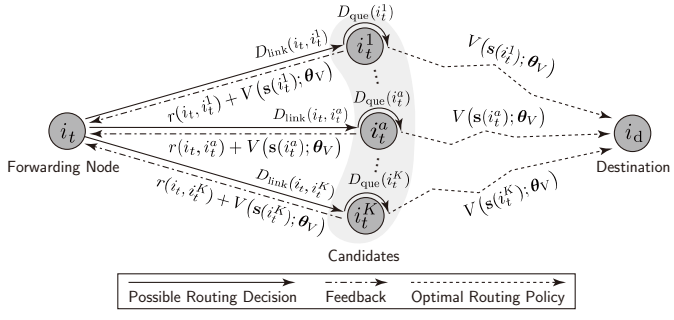


Fig. 1. Illustration of the feedback mechanism in DVN-routing.

Compared with directly determining the action based on the DQN by (7), the information used for deciding the action is observed by every next-hop candidate instead of that observed by the forwarding node alone. In this way, the forwarding node is able to plan one step ahead for more prompt adaption to the dynamic environment. For example, when the next-hop candidate i_t^a has a higher traffic load, the queuing delay $D_{\text{que}}(i_t^a)$ will increase, which increases the value of $r(i_t, i_t^a)$ and hence i_t^a is less likely to be chosen as the next hop according to (16).

The whole learning and decision procedure is shown in Algorithm 1.

Algorithm 1 DVN-Routing for AANETs

Initialize $\boldsymbol{\theta}_V$ and $\boldsymbol{\theta}'_V \leftarrow \boldsymbol{\theta}_V$.

Offline DVN Training

for episode $= 1, 2, \cdots, N$ **do**
  Randomly sample a topology snapshot from historical flight data.
  Set the source i_{s} and destination i_{d}.
  for $t = 1, 2, \cdots$ **do**
    if $i_t^{a_t} = i_{\text{d}}, t > t_{\max}$ **then**
      break
    Observe state $\mathbf{s}_t = \mathbf{s}(i_{\text{s}})$.
    Randomly select action $a_t \in \{1, \cdots, K\}$ (with probability ε), or set $a_t = \arg\min_a \left[r(i_t, i_t^a) + V\big(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V\big) \right]$ otherwise.
    Store the experience $\tilde{\mathbf{e}}_t$ composed by (14) into $\mathcal{D}$.
    Randomly sample a batch of experiences from $\mathcal{D}$ as $\mathcal{B}$.
    Update $\boldsymbol{\theta}_V$ and $\boldsymbol{\theta}'_V$ according to (15).

Online Routing Decision

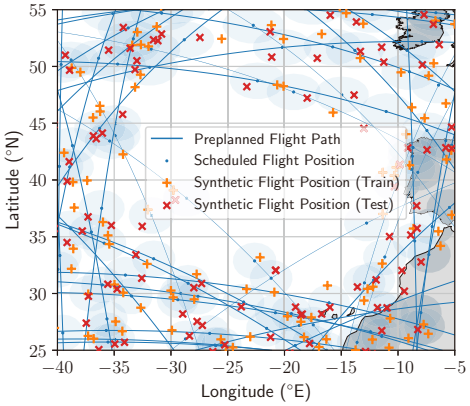
Input: i_{s}, i_{d}, $\boldsymbol{\theta}_V$.
for $t = 1, 2, \cdots$ **do**
  if $i_t = i_{\text{d}}$ **then**
    break
  The forwarding node i_t observes $\mathbf{s}(i_t)$.
  for $a = 1, \cdots, K$ **do**
    Node i_t^a observes $\mathbf{s}(i_t^a)$, estimates $r(i_t, i_t^a)$, computes $V(\mathbf{s}(i_t^a); \boldsymbol{\theta}_V) + r(i_t, i_t^a)$ and sends the result to node i_t.
  Node i_t computes a_t^* by (16) and forwards the packet to $i_t^{a_t^*}$.

IV. Simulation Results

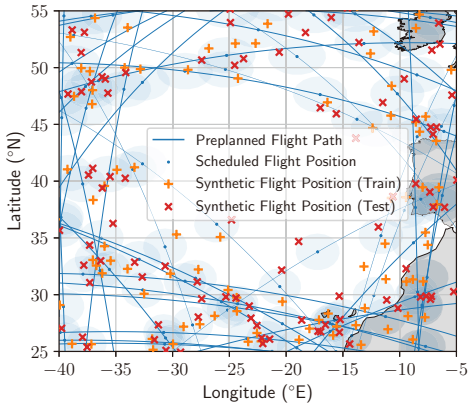
In this section, we introduce the simulation environment and compare the performance of the routing policies learned by DRL to benchmark policies via simulations.

A. Simulation Environment

Since there is insufficient real flight data available for training and testing, we generate synthetic flight data in our simulation for mimicking the airplane mobility. Specifically, we consider a 3D-airspace within longitude $-40° \sim -5°$ East, latitude $25° \sim 55°$ North, and altitude $0 \sim 13$ km, whose 2D-projection is shown in Fig. 2. To reflect the typical non-uniform flight density distribution and to evaluate the performance of routing algorithms when communication void exists, we earmark a pair of no-fly zones located at $(-17.25°, 40°, 0)$ and $(-27.75°, 44.5°, 0)$ each having a radius of 500 km and a height of 13 km, where no flight path passes through.



(a) Flight positions at time t_1



(b) Flight positions at time t_{50}

Fig. 2. Illustration of two example snapshots in training and testing set at different time. The shaded area represents the specific zone, where a particular flight may be found with probability 80%.

There are 40 preplanned great-circle flight paths randomly drawn through the available area and then fixed throughout the simulation to represent the seasonal flight corridors. A total of 100 airplanes are uniformly placed along the 40 flight paths, where the airplanes on the same path are flying in the same direction with constant speed to maintain the safety flight separation distance. The altitude of each airplane is randomly chosen within the normal cruise altitude of $9 \sim 13$ km.

In reality, each airplane may not takeoff on time and may not fly strictly according its preplanned path due to varies reasons, which results in the mismatch between the historical flight positions (i.e., training data) and current flight positions (i.e., testing data). To reflect this issue, we add a random deviation to the *scheduled flight position* (i.e., the position of airplane when it fly strictly according to the plan) to generate the *synthetic flight position* for training and testing the routing algorithm, as shown in Fig. 2. Specifically, the random deviations along the latitude and longitude follow Gaussian distribution with a standard deviation of 100 km.

We generate 2000 snapshots of the network, half of which are used for training and the other half are used for testing outside the training set. In Fig. 2, we demonstrate a pair of example snapshots at different time. We can see that the flight positions change over time, and the positions of the same flight in training and testing set are different.

In each snapshot, the source node is randomly selected, while the destination is set as the ground station is located at

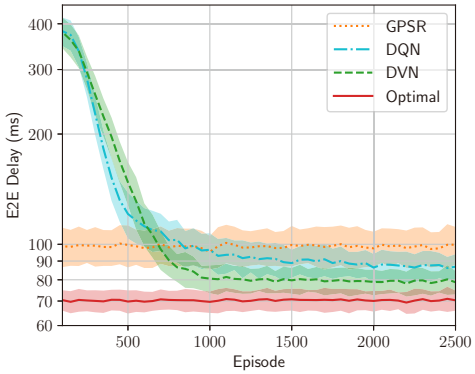


Fig. 3. Learning curve in training phase. All experiments are run for 100 different random seeds each. The curves are smoothed by averaging over a window of 40 episodes. The lines reflect the average value and the shaded bands reflect the standard deviation.

$(-10°, 52°, 0.05)$. The queuing delay is set as $D_{\text{que}} = 5$ ms throughout the training. The packet size is $S = 15$ KB and the transmission data rate is configured according to the distance-based adaptive coding and modulation scheme of [14, Table I] using matched filter based beamforming relying on 32 transmit antennas and four receive antennas.

B. Fine-Tuned Parameters of DQN- and DVN-Routing

The DNNs are tuned as follows to achieve their best performance. The candidate set size is $K = 10$. Both the DQN and DVN have two hidden layers, where each layer has 100 and 50 nodes for DQN and DVN, respectively. In this setting, the total number of unknown parameters to be learned in DVN is roughly reduced roughly by a factor of three compared to DQN. The hidden layers employ the rectified linear units (ReLU) as the activation function while the output layer has no activation function. In the training phase, the exploration probability is set as $\varepsilon = 1$ for the first 100 episodes, decreases to 0.1 within the next 400 episodes, and remains 0.1 for the rest of the episodes. The learning rate δ is 10^{-4} for both the DQN and DVN, while the update rate is $\tau = 10^{-3}$ for both the target networks. The batch size is $|\mathcal{B}| = 32$. During the testing phase, we set $\varepsilon = 0$ for both DQN and DVN, and the parameters $\boldsymbol{\theta}_V$, $\boldsymbol{\theta}_Q$ are frozen.

C. Performance Comparison

The following benchmarks are considered for comparison:

- Optimal: The optimal route found by solving problem (2) via the Floyd-Warshall algorithm, which relies on the global information regarding the link delay between every two nodes and the queuing delay of every single node.
- GPSR: The routing protocol proposed in [7], which is solely based on local geographic information. Specifically, each node forwards its received packet to the specific neighbor that is geographically closest to the destination. When a packet reaches a node where greedy forwarding fails, the algorithm recovers by routing around the perimeter of the region.

In Fig. 3, we compare the learning curves of the proposed DQN-routing and DVN-routing algorithms during training. We can see that both the algorithms achieve lower average E2E delay than GPSR after 500 episodes of training and finally

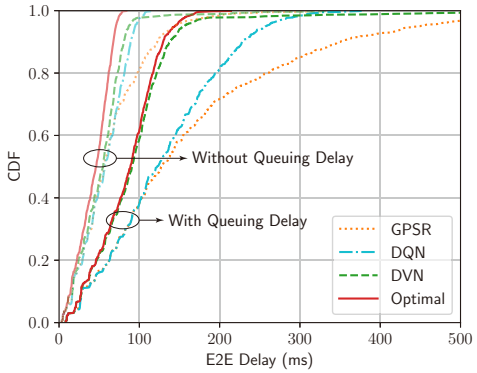


Fig. 4. The CDF of E2E delay with or without considering the queuing delay.

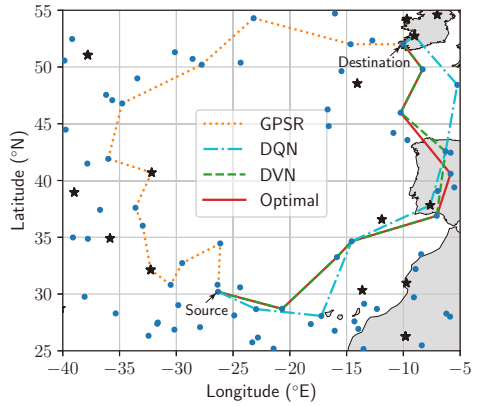


Fig. 5. The routes found by each routing algorithms in a example snapshot, where "★" denotes the congested nodes.

approach the delay of optimal routing. Furthermore, since DVN-routing exploits the knowledge concerning the system's dynamics, it achieves lower E2E delay than DQN-routing after its convergence.

In the online testing phase, the snapshots are generated outside the training set as previously mentioned to reflect the uncertainty in flight positions. Furthermore, to reflect the fluctuation of traffic load, 20% of the nodes are randomly chosen to set with a higher queuing delay of 50 ms. In Fig. 4, we compare the cumulative distribution function (CDF) curves of the E2E delay during the testing phase. We can see that upon neglecting the queuing delay in the E2E delay calculation, DQN achieves near-optimal performance. However, when taking the queuing delay into consideration, the gap between the optimal routing policy and DQN-routing increases. Nevertheless, DQN-routing still outperforms GPSR. By contrast, DVN-routing can still achieve near-optimal performance, even though it is trained offline assuming constant and identical queuing delay, because the feedback mechanism allows each next-hop candidate to report its real-time queuing delay to the forwarding node during the online decision phase.

In Fig. 5, we show an example snapshot during the testing phase for comparing the routes found by different routing algorithms. We can see that GPSR is rather "shortsighted" and struggles to get round the no-fly zone. By contrast, both DQN-routing and DVN-routing can find routes having a similar number of hops as the optimal routing policy, because they implicitly exploit the network topology information that has been embedded in the DQN/DVN trained using historical flight trajectories. Since DQN-routing is unaware of the real-time queuing delay, it may encounter some congested nodes (marked by "★") along the route. By arranging for each next-hop candidate to feed back its queuing delay to the forwarding node, DVN-routing can find a route bypassing the congested relaying nodes.

V. Conclusions and Future Directions

In this paper, we proposed DRL-based routing policies for minimizing the E2E delay in AANETs. We first used DQN for learning a direct mapping from the local geographic information to the optimal routing decision. To boost the learning efficiency and the online adaptability of the proposed DQN-routing, we additionally proposed DVN-routing by exploiting the knowledge concerning the system's dynamics and by introducing a feedback mechanism. Simulation results show that both DQN-routing and DVN-routing achieve lower E2E delay than GPSR, while DVN-routing performs very closely to the optimal routing based on global information.

It is worth noting that although AANETs can be formed in many regions where the flight-density is high enough, it may fail in certain regions where the flight-density is low. Future research may integrate low earth orbit satellites into the AANET for supporting truly global coverage.

References

[1] X. Huang, J. A. Zhang, R. P. Liu, Y. J. Guo, and L. Hanzo, "Airplane-aided integrated networking for 6G wireless: Will it work?" *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 84–91, Sept. 2019.

[2] J. Zhang, T. Chen, S. Zhong, J. Wang, W. Zhang, X. Zuo, R. G. Maunder, and L. Hanzo, "Aeronautical ad hoc networking for the Internet-above-the-clouds," *Proc. IEEE*, vol. 107, no. 5, pp. 868–911, May 2019.

[3] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Veh. Technol. Mag.*, vol. 2, no. 2, pp. 12–22, Jun. 2007.

[4] E. Sakhaee and A. Jamalipour, "The global in-flight Internet," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 9, pp. 1748–1757, Sept. 2006.

[5] Q. Luo and J. Wang, "Multiple QoS parameters-based routing for civil aeronautical ad hoc networks," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 804–814, Jun. 2017.

[6] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Netw.*, vol. 15, no. 6, pp. 30–39, Nov.–Dec. 2001.

[7] B. Karp and H.-T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM MobiCom*, 2000, pp. 243–254.

[8] D. Medina, F. Hoffmann, F. Rossetto, and C.-H. Rokitansky, "A geographic routing strategy for north Atlantic in-flight Internet access via airborne mesh networking," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 1231–1244, Aug. 2011.

[9] S. Wang, C. Fan, C. Deng, W. Gu, Q. Sun, and F. Yang, "A-GR: A novel geographical routing protocol for AANETs," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 931–937, Nov. 2013.

[10] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[11] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. NIPS*, 1994, pp. 671–678.

[12] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access*, vol. 7, pp. 55 916–55 950, 2019.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[14] J. Zhang, S. Chen, R. G. Maunder, R. Zhang, and L. Hanzo, "Adaptive coding and modulation for large-scale antenna array-based aeronautical communications in the presence of co-channel interference," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1343–1357, Feb. 2017.