Deep Learning Aided Packet Routing in Aeronautical *Ad-Hoc* Networks Relying on Real Flight Data: From Single-Objective to Near-Pareto Multi-Objective Optimization

Dong Liu, Jiankang Zhang, Jingjing Cui, Soon-Xin Ng, Robert G. Maunder, 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 learning (DL) to assist routing in AANETs. We set out from the single objective of minimizing the end-to-end (E2E) delay. Specifically, a deep neural network (DNN) is conceived for mapping the local geographic information observed by the forwarding node into the information required for determining the optimal next hop. The DNN is trained by exploiting the regular mobility pattern of commercial passenger airplanes from historical flight data. After training, the DNN is stored by each airplane for assisting their routing decisions during flight relying solely on local geographic information. Furthermore, we extend the DL-aided routing algorithm to a multi-objective scenario, where we aim for simultaneously minimizing the delay, maximizing the path capacity and maximizing the path lifetime. Our simulation results based on real flight data show that the proposed DL-aided routing outperforms existing position-based routing protocols in terms of its E2E delay, path capacity as well as path lifetime, and it is capable of approaching the Pareto front that is obtained using global link information.**

***Index Terms*—AANET, routing, deep learning, multi-objective optimization**

I. Introduction

Next-generation wireless networks are envisaged to support truly 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, 3].

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 routing path at any time [4]. Traditional topology-based *ad-hoc* routing protocols [5], such as the *ad-hoc* on-demand distance vector (AODV) [6] based solutions, 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 substantial research efforts have been invested in improving the stability of routing in AANETs [7, 8], they have a limited ability to update their routing tables for prompt adaptation in high-dynamic scenarios. Reinforcement learning based routing algorithms, such as Q-routing [9] and its variants [10] were proposed for improving the adaptability of routing in dynamic environments. However, they require frequent information exchange for trail-and-error in order to update the Q-table (acting as the routing table) in an online manner, which still suffer from the above issues when the network topology is highly dynamic.

By contrast, another family of *ad-hoc* routing protocols, namely position-based (or geographic) routing [15], 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 for determining the next hop can be readily obtained by each airplane using the automatic dependent surveillance-broadcast (ADS-B) system on board, geographic routing is more appealing in AANETs. Greedy perimeter stateless routing (GPSR) [11] was one of the most popular geographic routing protocols, which has also inspired various extensions [12, 13] 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 [12], greedy routing was improved for avoiding congestion by considering the buffer queue status of the next-hop candidates. In [13], the mobility information was further taken into consideration for choosing a more stable next hop. However, greedy routing [11–13] is stifled when no neighbor is closer to the destination than the forwarding node (this situation is term as the *communication void*).

To elaborate, the limitation of position-based routing arises from the fact that the nodes are unaware of the entire

D. Liu is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (email: dliu@buaa.edu.cn).

J. Cui, S. X. Ng, R, G. Maunder, and L. Hanzo are with the School of Electronics and Computer Science, the University of Southampton, Southampton SO17 1BJ, UK (email: jingj.cui@soton.ac.uk; sxn@ecs.soton.ac.uk; rm@ecs.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).

L. Hanzo would like to acknowledge the financial support of the Engineering and Physical Sciences Research Council projects EP/P034284/1 and EP/P003990/1 (COALESCE) as well as of the European Research Council's Advanced Fellow Grant QuantCom (Grant No. 789028)

TABLE I
COMPARISON WITH EXISTING ROUTING ALGORITHMS FOR AANETS

Routing Algorithms	AODV [6] MUDOR [7]	GPSR [11] GLSR [12] A-GR [13]	Q-Routing [9] & its variants [10]	MQSPR [8]	MOEA [14]	Proposed POMOR	Proposed DL-aided Routing
Centralized: Global Topology Information Low-Dynamic Scenario					√	√	
Ad-Hoc (Topology-Based): Routing Table Low-Dynamic Scenario	√		√	√			
Ad-Hoc (Position-Based): Local Geographic Information High-Dynamic Scenario		√					√
Single Objective	√	√	√				√
Mutliple Objective				√	√	√	√
Find All Pareto-Optimal Paths						√	

network topology. Therefore, one of our ambitious goals is to *enable the forwarding node to infer the information required for avoiding the communication void issue from its local observation*. Although the topology of AANETs evolves dynamically, it exhibits certain patterns, since both the flight trajectories and takeoff times are pre-planned and normally repeated on a weekly basis. This suggests that the local geographic information may be strongly correlated with that of the whole topology, hence this correlation may be exploited using historical flight data.

Another typical limitation of the existing routing protocols designed for AANETs is that only a single-component objective function (OF) is optimized for improving a specific network performance metric. However, the overall network performance should be characterized by multiple metrics, such as the end-to-end (E2E) packet delay, path capacity and path lifetime. Accordingly, multi-objective optimization (MOO) [16] can be adopted, where all components of the OF are optimized simultaneously. Since multiple objectives typically conflict with each other, they may not achieve their respective optima at the same time. Hence, in contrast to optimizing a single-component OF, typically there is no single globally optimal solution in MOO, which is the best with respect to all objectives. By contrast, the ultimate goal of MOO is to discover all the *Pareto-optimal* solutions that constitute the entire *Pareto front*, where none of the objective can be improved without scarifying at least one of the other objectives [16]. To expound a little further, the Pareto front characterizes the optimal tradeoff relationship among multiple potentially conflicting objectives. As a benefit, the network operator may opt for any of the Pareto-optimal solutions along the Pareto front according to the requirements of different services. For example, for file downloading, it is more important to increase the capacity than reducing the E2E delay, while for voice calls, it is more important to reduce the E2E delay.

MOO has been leveraged in diverse problems in wireless networks [17, 18] and also been used for routing in AANET [8]. However, multiple performance metrics were combined by a linear weighted sum in [8], by which some Pareto-optimal solutions may not be found, when the Pareto front is non-convex. Alternatively, bio-inspired metaheuristics, such as multi-objective evolutionary algorithms (MOEAs), can be leveraged for solving the multi-objective routing problem [14], which have the potential to find the Pareto-optimal solutions even in non-convex scenarios. However, MOEAs require global knowledge regarding the status (e.g., delay, capacity, lifetime) of every single possible link in the network for running the optimization, which is not feasible in large-scale AANETs.

Therefore, another radical goal of this paper is to *devise routing algorithm simultaneously optimizing multiple performance metrics based on local information in a distributed manner*. As we mentioned before, the local geographic information may be strongly correlated with the global topology due to the regular mobility pattern of commercial passenger airplanes. In this context, deep neural networks (DNNs) [19] are capable of learning a direct mapping from the input features to the desired output.

Against this background, we conceive a bespoke DL technique for learning routing in AANETs. We commence with a single objective aimed at minimizing the E2E delay and then extend our investigations to a multi-objective scenario, where the E2E delay, path capacity, and path lifetime are jointly considered as components of the OF. The contributions of this paper are summarized as follows:

1) We propose a DL-aided routing algorithm for minimizing the E2E delay in AANETs. Specifically, we use a DNN that takes the coordinates of the next-hop candidates as its input and outputs the information required for determining the optimal next hop. The DNN is trained offline by supervised learning, where the training labels are obtained by running the shortest path algorithm using historical flight data. With the aid of the pre-trained DNN, the optimal next hop can be determined promptly, solely based on local geographic information.
2) We further extend our DL-aided routing algorithm for handling multi-component OF, namely simultaneously minimizing the delay, maximizing the path capacity and maximizing the path lifetime. To generate training labels for the DNN, we propose a Pareto-optimal multi-

TABLE II
SUMMARY OF MAIN NOTATIONS

$\mathcal{N}^{[t]}$	Set of nodes available at TS t	p_n	The nth node on path $\boldsymbol{p}$
$\boldsymbol{s}_i^{[t]}, \bar{\boldsymbol{s}}_i^{[t]}$	Spherical/Cartesian coordinates of node i at TS t	$D^{[t]}(\boldsymbol{p})$	Delay of path $\boldsymbol{p}$ at TS t
$\theta_i^{[t]}$	Latitude of node i at TS t	$C^{[t]}(\boldsymbol{p})$	Capacity of path $\boldsymbol{p}$ at TS t
$\varphi_i^{[t]}$	Longitude of node i at TS t	$L^{[t]}(\boldsymbol{p})$	Lifetime of path $\boldsymbol{p}$ at TS t
$h_i^{[t]}$	Altitude of node i at TS t	$L^{[t]}(i,j)$	Lifetime of link $i \to j$ starting from TS t
R_{e}	Earth radius	$\boldsymbol{p}^*_{i_{\mathsf{s}}}$	Optimal path from node i_{s} to i_{d}
$d^{[t]}(i,j)$	Distance between nodes i and j at TS t	$D^{[t]}_*(j,i_d)$	Minimum delay from j to i at TS t
$d_{\mathsf{th}}(h_i^{[t]}, h_j^{[t]})$	Maximum distance of a direct link	$\hat{D}^{[t]}_*(j,i_d)$	Estimated minimum delay using DNN
$\mathcal{B}_i^{[t]}$	Neighbor set of node i at TS t	$\tilde{D}^{[t]}_*(j,i_d)$	Estimated minimum delay using feedback
$b_k^{[t]}(i)$	The kth neighbor of node i at TS t	$\boldsymbol{x}_i^{[t]}$	Local information observed by node i at TS t
$C^{[t]}(i,j)$	The capacity of link $i \to j$ at TS t	$\boldsymbol{y}_i^{[t]}, \hat{\boldsymbol{y}}_i^{[t]}$	Desired/Actual output of the DNN at node i
W, f	Transmission bandwidth/Carrier frequency	$\boldsymbol{f}(\cdot;\boldsymbol{\theta})$	DNN with parameter $\boldsymbol{\theta}$
P, σ^2	Transmit/noise power	$\mathcal{C}_i^{[t]}$	Next-hop candidate set of node i
$G_{\mathsf{t}}, G_{\mathsf{r}}$	Transmit/Receive antenna gain	$\mathcal{M}_i^{[t]}$	Mutual candidate set of node i, see (30)
$D^{[t]}(i,j)$	Total delay of link $i \to j$	m_k	The kth element in the sorted mutual candidate set
$D^{[t]}_{\mathsf{que}}(i)$	Queuing delay at node i	$\bar{\mathcal{C}}_i^{[t]}$	Defined in (35)
$D^{[t]}_{\mathsf{tr}}(i,j)$	Transmission delay of link $i \to j$	$X^{[t]}_*(j,i_{\mathsf{d}},\varepsilon_C,\varepsilon_L)$	Defined in (42)
$D^{[t]}_{\mathsf{pr}}(i,j)$	Propagation delay of link $i \to j$	$\hat{X}(j,i_{\mathsf{d}},\varepsilon_C,\varepsilon_L)$	The estimate of $X^{[t]}_*(j,i_{\mathsf{d}},\varepsilon_C,\varepsilon_L)$
S	Packet size	$\tilde{X}(j,i_{\mathsf{d}},\varepsilon_C,\varepsilon_L)$	The estimate of $X^{[t]}_*(j,i_{\mathsf{d}},\varepsilon_C,\varepsilon_L)$ using feedback

objective routing (POMOR) algorithm using the global link information to obtain all the Pareto-optimal solutions at a moderate polynomial complexity.

3) Our simulation results based on real flight data collected over the North Atlantic Ocean and the European Continent show that:
   - For minimum-delay routing, our DL-aided routing outperforms the existing geographic routing and it is capable of approaching the performance of the optimal routing that relies on global link information.
   - For multi-objective routing, our proposed POMOR algorithm using global link information can find the Pareto front for visualizing and analyzing the optimal tradeoff between multiple performance metrics. When relying solely on local information, our DL-aided routing is capable of discovering paths that achieve a performance closely the Pareto front obtained by POMOR.

In Table I, we boldly and explicitly contrast our work to the most pertinent routing algorithms designed for AANETs. The rest of the paper is organized as follows. In Section II, we introduce the system model. In Section III, we propose our DL-aided routing algorithm for single objective optimization minimizing the E2E delay. In Section IV, we extend the DL-aided routing to a challenging multi-objective scenario. Our simulations results are provided in Section V, and finally, Section VI concludes the paper.

II. System Model

Consider an AANET formed by multiple passenger airplanes and a ground station (GS). Let i denote the node's index and $\mathcal{N}^{[t]}$ denote the set of node indices at timestamp (TS) t. The position of node i at TS t is denoted by a 3D-vector $\boldsymbol{s}_i^{[t]} = \left[\theta_i^{[t]}, \varphi_i^{[t]}, h_i^{[t]}\right]$ where $\theta_i^{[t]}$, $\varphi_i^{[t]}$, and $h_i^{[t]}$ denote the latitude, longitude, and altitude of node i at TS t, respectively. The main notations to be used throughout the article are summarized in Table. II.

In Cartesian coordinates, the position of node i can be converted as

$$\bar{\boldsymbol{s}}_i^{[t]} = \Big[(R_{\mathsf{e}} + h_i^{[t]})\cos\theta_i^{[t]}\cos\varphi_i^{[t]}, (R_{\mathsf{e}} + h_i^{[t]})\cos\theta_i^{[t]}\sin\varphi_i^{[t]}, (R_{\mathsf{e}} + h_i^{[t]})\sin\varphi_i^{[t]}\Big] \tag{1}$$

where $R_{\mathsf{e}} = 6371$ km is the earth radius. Then, the Euclidean distance between node i and node j at TS t can be expressed as $d^{[t]}(i,j) = \left\|\bar{\boldsymbol{s}}_i^{[t]} - \bar{\boldsymbol{s}}_j^{[t]}\right\|$.

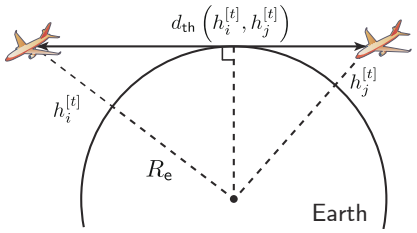


Fig. 1. The maximum distance for direct communication between two nodes above the horizon.

A pair of nodes can establish direct communications when they have direct visibility, i.e., when they are both above the horizon, as shown in Fig. 1. Then, the maximum communication distance between nodes i and j is given by

$$d_{\mathsf{th}}\left(h_i^{[t]}, h_j^{[t]}\right) = \sqrt{(h_i^{[t]} + R_{\mathsf{e}})^2 - R_{\mathsf{e}}^2} + \sqrt{(h_j^{[t]} + R_{\mathsf{e}})^2 - R_{\mathsf{e}}^2}. \tag{2}$$

Therefore, the condition that nodes i and j can establish a direct communication link is $d^{[t]}(i,j) \le d_{\mathsf{th}}(h_i^{[t]}, h_j^{[t]})$.

Let $\mathcal{B}_i^{[t]} = \{j | d^{[t]}(i,j) \le d_{\mathsf{th}}(h_i^{[t]}, h_j^{[t]})\}$ denote the set of nodes that can connect to node i, i.e., the *neighbors* of node i. Moreover, we rank the neighbors by their distance to the destination in ascending order and let $b_k^{[t]}(i)$ denote the kth neighbor of node i at TS t.

Upon assuming free-space path loss above the clouds, the capacity of the direct link from node i to node j can be

expressed as

$$C^{[t]}(i,j) = W \log_2 \left[1 + \frac{P G_{\mathsf{t}} G_{\mathsf{r}}}{\sigma^2} \left(\frac{c}{4\pi f d^{[t]}(i,j)} \right)^2 \right], \quad (3)$$

where W is the transmission bandwidth, P is the transmit power, G_{t} and G_{r} denote the transmit and receive antenna gains, respectively, σ^2 is the noise power, $c = 3 \times 10^8$ m/s is the speed of light, and f is the carrier frequency.

Let S denote the packet size. The delay of sending a packet from node i to node j is composed by

$$D^{[t]}(i,j) = D^{[t]}_{\mathsf{que}}(i) + D^{[t]}_{\mathsf{tr}}(i,j) + D^{[t]}_{\mathsf{pr}}(i,j), \quad (4)$$

where $D^{[t]}_{\mathsf{que}}(i)$ denotes the duration that the packet spent in the buffer queue of node i, which can be formulated as $D^{[t]}_{\mathsf{que}}(i) = \sum_{q=1}^{Q_i^{[t]}} \frac{S}{C^{[t]}(i,n_q)}$ with $Q_i^{[t]}$ denoting the number of packets in the buffer queue of node i and n_q denoting the next hop for the qth packet in the queue, $D^{[t]}_{\mathsf{tr}}(i,j) = \frac{S}{C^{[t]}(i,j)}$ is the transmission delay, and $D^{[t]}_{\mathsf{pr}}(i,j) = \frac{d^{[t]}(i,j)}{c}$ is the propagation delay from node i to node j.

Since the distance between a source node i_{s} and a destination node i_{d} may exceed the direct communication range, the packet may traverse through multiple nodes until finally reaching i_{d}. Let $\boldsymbol{p} \triangleq (p_1, p_2, \cdots, p_N)$ denote a path having $N-1$ hops connecting nodes i_{s} and i_{d}, where p_n denotes the index of the nth node on path $\boldsymbol{p}$. In particular, we have $p_1 = i_{\mathsf{s}}$ and $p_N = i_{\mathsf{d}}$.

We consider three metrics for routing path selection. The first one is the E2E delay of the path, which is the summation over the delay of each link along the path and can be formulated as

$$D^{[t]}(\boldsymbol{p}) = \sum_{n=1}^{N-1} D^{[t]}(p_n, p_{n+1}). \quad (5)$$

The second one is the path capacity, which is limited by the lowest link capacity along the path as

$$C^{[t]}(\boldsymbol{p}) = \min_{n=1,\cdots,N-1} C^{[t]}(p_n, p_{n+1}). \quad (6)$$

Since AANETs have dynamic typologies, a communication link breaks when the transmitter or receiver move out of each other's communication range. To reflect routing stability, we consider the path lifetime as the third metric, which is defined as the duration when every transmitter-receiver pair along the path has direct visibility. A long path lifetime reflects a more stable path, avoiding rerouting. Specifically, the lifetime of path $\boldsymbol{p}$ starting from TS t can be formulated as

$$L^{[t]}(\boldsymbol{p}) = \min_{n=1,\cdots,N-1} L^{[t]}(p_n, p_{n+1}). \quad (7)$$

where $L^{[t]}(i,j)$ denote the lifetime of link $i \to j$, which is the duration of transmitter i and receiver j being within the communications range, i.e.,

$$L^{[t]}(i,j) = \max_{\Delta t} \left\{ \Delta t \,\middle|\, d^{[t+\Delta t]}(i,j) \le d_{\mathsf{th}} \left(h_i^{[t+\Delta t]}, h_j^{[t+\Delta t]} \right) \right\}. \quad (8)$$

We can observe that the link lifetime $L^{[t]}(i,j)$ depends on the future distance between i and j in each time stamp, which further depends on the coordinates of nodes i and j in each future TS, i.e., $\left\{ \boldsymbol{s}_i^{[t+\Delta t]} \right\}_{t \ge 0}$ and $\left\{ \boldsymbol{s}_j^{[t+\Delta t]} \right\}_{t \ge 0}$. In practice, the future positions of an airplane can be obtained from its pre-planned trajectory.[1] Then, the distance $d^{[t+\Delta t]}(i,j)$ in each future TS can be calculated for checking whether the maximum communication range is reached, and hence the link lifetime $L^{[t]}(i,j)$ can be calculated effortlessly, by evaluating (8) using the classic bi-section search method.

III. Single-Objective Routing

In this section, we first formulate a single-objective routing problem aimed at minimizing the E2E delay and then propose the corresponding DL-aided routing algorithm.

A. Problem Formulation

Let $\boldsymbol{p}^*_{i_{\mathsf{s}}}$ denote the optimal path minimizing the E2E delay between an arbitrary source node i_{s} and the destination node i_{d} at an arbitrary TS t. The routing optimization problem minimizing the E2E delay can be formulated as

$$\mathsf{P}_D^{[t]}: \quad \min_{\boldsymbol{p}} \; D^{[t]}(\boldsymbol{p}) = \sum_{n=1}^{N-1} D^{[t]}(p_n, p_{n+1}) \quad (9a)$$

$$s.t. \; d^{[t]}(p_n, p_{n+1}) \le d_{\mathsf{th}}(h^{[t]}_{p_n}, h^{[t]}_{p_{n+1}}), \quad \forall n = 1, \cdots, N-1 \quad (9b)$$

$$p_n \in \mathcal{N}^{[t]}, \; \forall n = 1, \cdots, N \quad (9c)$$

$$p_1 = i_{\mathsf{s}}, \; p_N = i_{\mathsf{d}}. \quad (9d)$$

which can be regarded as a shortest path problem by treating the link delay as the "distance". However, conventional graph-based methods, such as Dijkstra's algorithm, are not applicable in practice due to the following reasons. To solve a shortest path problem using Dijkstra's algorithm, the global knowledge regarding the graph is required. Specifically, the whole network topology, the propagation and transmission delay between every pair of airplanes, and the queuing delay at each airplane should be known. This requires a centralized controller collecting all the required information, which goes against the self-organized nature of AANETs. Moreover, due to the high-dynamic network topology of AANETs, the minimum-delay path changes rapidly over time, and hence Dijkstra's algorithm has to be re-run frequently in order to keep the minimum-delay path up-to-date. Consequently, the required global information should be frequently refreshed to the controller, which imposes excessive signaling overhead. Therefore, our goal is to solve problem $\mathsf{P}_D^{[t]}$ relying solely on local information in a distributed manner.

In the following, we assume that each node is aware of its own position, the positions of its neighbors as well as of the destination, and then invoke DL for finding the optimal path.

B. DL-Aided Minimum Delay Routing Using Local Information

1) Optimal Substructure: Problem $\mathsf{P}_D^{[t]}$ has an optimal substructure as shown in Fig. 2. Specifically, assume that the

[1]When there is no pre-planned trajectory, the future positions can be predicted based on the current coordinates, speeds and headings of nodes i and j, assuming that they fly at a constant altitude, speed as well as heading.

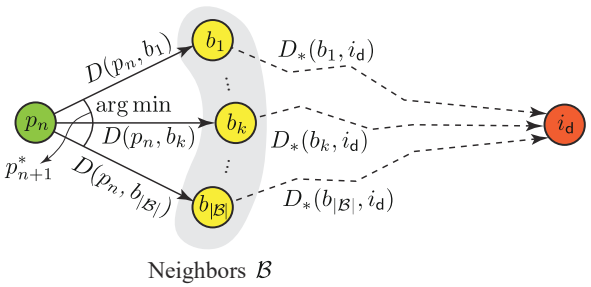


Fig. 2. Finding the optimal next hop using the optimal substructure. A straight line indicates a direct communication link and a polyline indicates a minimum-delay path between the two nodes it connects. For notational simplicity, we omit subscript and superscript of "$\mathcal{B}_{p_n}^{[t]}$", and the parentesis of "$b_k(p_n)$"

packet is currently located at node p_n. Let $j \in \mathcal{B}_{p_n}^{[t]}$ denote the index of p_n's neighbor and let $D_*^{[t]}(j, i_\mathsf{d})$ denote the minimum delay from node j to the destination i_d. Then, the optimal next hop minimizing the overall delay from the current forwarding node p_n to the destination i_d can be characterized by

$$p^*_{n+1} = \arg \min_{j \in \mathcal{B}_{p_n}^{[t]}} \left\{ D^{[t]}(p_n, j) + [1 - I_\mathsf{d}(j)] D_*^{[t]}(j, i_\mathsf{d}) \right\}, \tag{10}$$

where $I_\mathsf{d}(j) = 1$ if $j = i_\mathsf{d}$, and $I_\mathsf{d}(j) = 0$ otherwise. Relying on the optimal substructure, the optimal next hop can be determined when $D^{[t]}(p_n, j)$ and $D_*^{[t]}(j, i_\mathsf{d})$ for $j \in \mathcal{B}_{p_n}^{[t]}$ are known. Consequently, the optimal path $\boldsymbol{p}^*_{i_\mathsf{s}}$ can be obtained by determining the optimal next hop one by one until the packet reaches its destination i_d.

Since $D^{[t]}(p_n, j)$ can be readily measured by the forwarding node, the optimal next hop can be determined once the minimum delay $D_*^{[t]}(j, i_\mathsf{d})$ from each neighbor $j \in \mathcal{B}_{p_n}^{[t]}$ to the destination i_s is obtained. Recalling that the goal is to determine the optimal next hop relying solely on local information, it can be accomplished if we can infer $D_*^{[t]}(j, i_\mathsf{d})$ from local information. In practice, $D_*^{[t]}(j, i_\mathsf{d})$ depends on the global network topology, which is strongly correlated with the local topology due to the regular flight trajectory and schedule. This motivates us to use a DNN for learning the relationship between the local geographic information observed by the forwarding node and the minimum delay from each neighbor to the destination as follows.

2) Structure of the DNN: We first specify the input and output of the DNN. To reflect the local topology observed by the forwarding node i, the input feature of the DNN should naturally include the position of node i and the positions of its neighbors $\mathcal{B}_i^{[t]}$. However, the number of neighbors is in general different for each node $i \in \mathcal{N}^{[t]}$, while the DNN's input dimension is fixed. If each forwarding node sends its packets to a neighbor that is far away from the destination, then the overall number of hops required for reaching the destination become excessive. Moreover, the computational complexity increases with the number of neighbors. Therefore, it is reasonable to rank the neighbors according to their geographic distance from the destination in ascending order and only consider the top-K ranked neighbors, i.e., $b_1(i), \cdots, b_K(i)$. By further including the position of the destination i_d, the input feature of the DNN can be finally formulated as

$$\boldsymbol{x}_i^{[t]} = \left[\boldsymbol{s}_i^{[t]}, \boldsymbol{s}_{b_1(i)}^{[t]}, \cdots, \boldsymbol{s}_{b_K(i)}^{[t]}, \boldsymbol{s}_{i_\mathsf{d}}^{[t]} \right] \in \mathbb{R}^{3(K+2)}, \tag{11}$$

Since the number of neighbors may be lower than K for some nodes, we manually set $\boldsymbol{s}_{b_k(i)}^{[t]} = \boldsymbol{0}$ if $k > \left|\mathcal{B}_i^{[t]}\right|$.

Recall that the DNN is used for learning the relationship between the local geographic information and the minimum delay from each neighbor to the destination. Therefore, the desired output of the DNN is designed as

$$\boldsymbol{y}_i^{[t]} = \left[D_*^{[t]}(b_1(i), i_\mathsf{d}), \cdots, D_*^{[t]}(b_K(i), i_\mathsf{d}) \right] \in \mathbb{R}^K, \tag{12}$$

where again, we only consider the top-K ranked neighbors for fixing the output dimension of the DNN and manually set $D_*^{[t]}(b_k(i), i_\mathsf{d}) = \mathsf{inf}$[2] for $k > \left|\mathcal{B}_i^{[t]}\right|$.

The structure of the DNN is shown in Fig. 3. Since each dimension of the input has different units, e.g. [°] for the longitude and latitude while [km] for the altitude, we use batch normalization [20] for normalizing each dimension across the samples in a mini-batch to have zero mean and unit variance.

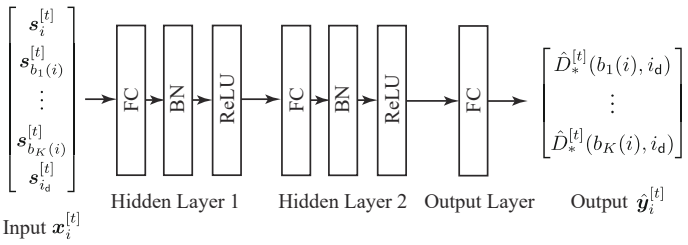


Fig. 3. DNN architecture for minimum delay routing, where "FC" represents the fully-connected layer, "BN" stands for batch normalization, and "ReLU" is the rectified linear unit.

Let $\boldsymbol{\theta}$ denote the collection of unknown parameters to be trained. The DNN can be expressed as a function of $\boldsymbol{x}_i^{[t]}$ with parameter $\boldsymbol{\theta}$ as $\boldsymbol{f}(\boldsymbol{x}_i^{[t]}; \boldsymbol{\theta})$, whose output is the estimate of $\boldsymbol{y}_i^{[t]}$ formulated as

$$\hat{\boldsymbol{y}}_i^{[t]} = \boldsymbol{f}(\boldsymbol{x}_i^{[t]}; \boldsymbol{\theta}) = \left[\hat{D}_*^{[t]}(b_1(i), i_\mathsf{d}), \cdots, \hat{D}_*^{[t]}(b_K(i), i_\mathsf{d}) \right], \tag{13}$$

where $\hat{D}_*^{[t]}(b_k(i), i)$ denotes the estimate of $D_*^{[t]}(b_k(i), i)$, and again, we manually set $\hat{D}_*^{[t]}(b_k(i), i_\mathsf{d}) = \mathsf{inf}$ for $k > \left|\mathcal{B}_i^{[t]}\right|$.

The expression of $\boldsymbol{f}(\cdot; \boldsymbol{\theta})$ is determined by cascading the expression of each layer in the DNN as

$$\hat{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{f}_K \left(\cdots \left(\boldsymbol{f}_k \left(\cdots \boldsymbol{f}_1 \left(\boldsymbol{x}; \boldsymbol{\theta}_1 \right) \cdots \right); \boldsymbol{\theta}_k \right) \cdots; \boldsymbol{\theta}_K \right), \tag{14}$$

where $\boldsymbol{f}_k(\cdot; \boldsymbol{\theta}_k)$ represents the kth layer of the DNN with parameter $\boldsymbol{\theta}_k$.

Specifically, the relationship between the output $\boldsymbol{y}_\mathsf{out}$ and input $\boldsymbol{x}_\mathsf{in}$ of the fully-connected layer is expressed as

$$\boldsymbol{y}_\mathsf{out} = \boldsymbol{f}_\mathsf{FC}(\boldsymbol{x}_\mathsf{in}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{W}\boldsymbol{x}_\mathsf{in} + \boldsymbol{b}, \tag{15}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ denote the weight and bias parameters, respectively. The batch normalization layer can be represented

[2]inf is defined as a very large number, e.g., $\mathsf{inf} = 10^8$.

by

$$\boldsymbol{y}_{\mathsf{out}} = \boldsymbol{f}_{\mathsf{BN}}(\boldsymbol{x}_{\mathsf{in}}; \boldsymbol{\gamma}, \boldsymbol{\beta}) = \boldsymbol{\gamma} \circ \frac{\boldsymbol{x}_{\mathsf{in}} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} + \boldsymbol{\beta}, \tag{16}$$

where "$\circ$" denotes the element-wise multiplication, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are the scale and shift parameters, respectively, and ϵ is a small constant introduced for ensuring numerical stability. During training, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ represent the empirical mean and variance of the input mini-batch, respectively. During testing (i.e., inference), $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ represent the population mean and variance of the entire training set [20]. The rectified linear unit (ReLU) layer implements the following operation

$$\boldsymbol{y}_{\mathsf{out}} = \boldsymbol{f}_{\mathsf{ReLU}}(\boldsymbol{x}_{\mathsf{in}}) = \max\{\boldsymbol{x}_{\mathsf{in}}, \boldsymbol{0}\}, \tag{17}$$

where the $\max$ operation is carried out in an element-wise manner.

3) Offline Training: Since the takeoff times as well as the trajectories of commercial passenger airplanes are pre-planned and normally repeated on a weekly basis, the flight data on the same day of different weeks are similar. Therefore, we can train the DNN based on historical flight data or on the pre-planned flight trajectories in an offline manner.

The goal of training is to minimize the loss function composed of the mean squared error between the actual output $\hat{\boldsymbol{y}}_i^{[t]} = \boldsymbol{f}(\boldsymbol{x}_i^{[t]}; \boldsymbol{\theta})$ and the desired output $\boldsymbol{y}_i^{[t]}$ of the DNN, yielding:

$$\mathsf{P}_{\mathsf{loss}}: \quad \min_{\boldsymbol{\theta}} \ \frac{1}{N_{\mathsf{total}}} \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N}^{[t]}} \left\| \boldsymbol{y}_i^{[t]} - \boldsymbol{f}(\boldsymbol{x}_i^{[t]}; \boldsymbol{\theta}) \right\|^2, \tag{18}$$

where $\mathcal{T}$ denotes the set of TSs of the historical flight data and N_{total} is the number of training samples.

In the following, we first generate the training samples, including the input $\boldsymbol{x}_i^{[t]}$ and the desired output $\boldsymbol{y}_i^{[t]}$ for $t \in \mathcal{T}$ and $i \in \mathcal{N}^{[t]}$. For each TS t in the historical dataset, we can retrieve the position of each node from the flight trajectories and create a snapshot of the whole network topology. Consequently, for each node $i \in \mathcal{N}^{[t]}$, we can obtain the DNN's input $\boldsymbol{x}_i^{[t]}$ from the position of each node.

To obtain the desired output $\boldsymbol{y}_i^{[t]}$ as the training label, we have to compute the minimum delay from each neighbor of node i to the destination node, i.e., $\hat{D}_*^{[t]}(j, i_{\mathsf{d}})$ for $\forall j \in \mathcal{B}_i^{[t]}$. Since the position of each node can be retrieved from the historical flight data, the transmission delay and propagation delay of each link can be readily calculated based on the expressions given after (4). As for the queuing delay, since we aim to train the DNN for exploiting the correlation between local and global network topology, which is independent from the traffic load, we assume that the queuing delay is identical and constant among all the nodes during training. In this way, the total queuing delay is determined by the number of hops in the path so that the network topology information is implicitly reflected. Then, the delay of every link in the network can be calculated by (4). Consequently, we can obtain the minimum delay $D_*^{[t]}(j, i_{\mathsf{d}})$ for $\forall j \in \mathcal{B}_i^{[t]}$ by setting $i_{\mathsf{s}} = j$ and solving problem $\mathsf{P}_D^{[t]}$ using any shortest path algorithm.

Upon obtaining the training set, we can train the DNN by solving problem $\mathsf{P}_{\mathsf{loss}}^{[t]}$ using stochastic gradient descent algorithms, such as the Adam method [21]. We note that the DNN is not designed and trained for a specific node. Instead, it can be used by all the forwarding nodes during the time window $\mathcal{T}$ in online testing. In fact, different forwarding nodes are distinguished by their local geographic information $\boldsymbol{x}_i^{[t]}$ and the desired DNN's input-output pairs of all nodes $\mathcal{N}_t$ during $\mathcal{T}$, i.e., $(\boldsymbol{x}_i^{[t]}, \boldsymbol{y}_i^{[t]})$ for all $i \in \mathcal{N}_t$ and $t \in \mathcal{T}$, are included in the training set. Therefore, once the DNN becomes sufficiently well-trained, all the forwarding nodes are able to obtain the required information for determining their own optimal next hop from the same DNN by inputting their own local geographic observation.

The benefit of all nodes using the same DNN is that it enables a certain degree of parameter sharing among different nodes, which improves the learning efficiency and scalability. For example, when two nodes are close, they share a similar set of neighbors and hence provide similar input for the DNN. Moreover, the outputs of their DNNs (i.e., the minimum delay from the neighbors to the destination) should also be similar, which suggests that the DNNs of these pair of nodes should be similar. This can be naturally achieved if the nodes share the same DNN.

4) On-line Decision Without Feedback: Once the DNN has become sufficiently well-trained, the DNN is copied to each airplane in support of online routing decisions.

Assume that the packet is currently located at node p_n. We first specify the next-hop candidates, namely the set of nodes where the next hop of p_n is chosen from. Since the input and output dimensions of the DNN are fixed, the next-hop candidates are restricted the top-K ranked neighbors of p_n.[3]

Moreover, to avoid loops in the path, the nodes that already exist in the path should be excluded. Therefore, the next-hop candidate set of p_n is specified as

$$\mathcal{C}_{p_n}^{[t]} = \{b_1(p_n), \cdots, b_K(p_n)\} \backslash \{p_1, \cdots, p_n\} \tag{19}$$

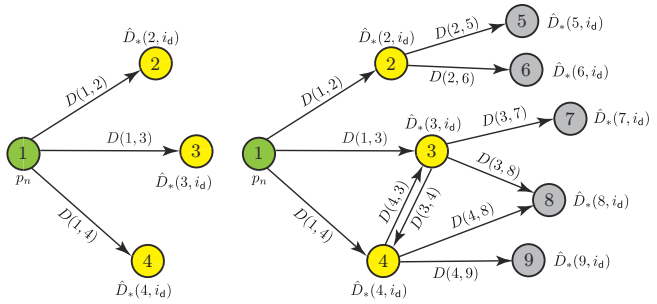
By observing the local geographic information, node p_n can calculate the delay from itself to each of its next-hop candidates, i.e., $D^{[t]}(p_n, j)$ for $j \in \mathcal{C}_{p_n}^{[t]}$, and gathers its input features $\boldsymbol{x}_{p_n}^{[t]}$ for the DNN. Then, from the output of the DNN, the forwarding node p_n can obtain the estimate of the minimum delay from each of its next-hop candidates to the destination, i.e., $\hat{D}_*^{[t]}(j, i_{\mathsf{d}})$ for $j \in \mathcal{C}_{p_n}^{[t]}$.

According to the optimal substructure (10), a straightforward way of determining the next hop using the DNN is by evaluating

$$p_{n+1} = \arg \min_{j \in \mathcal{C}_{p_n}^{[t]}} \left\{ D^{[t]}(p_n, j) + [1 - I_{\mathsf{d}}(j)] \, \hat{D}_*^{[t]}(j, i_{\mathsf{d}}) \right\}. \tag{20}$$

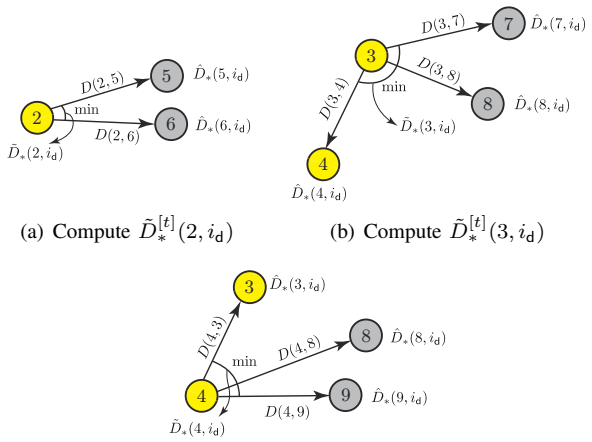
Recall that the training of DNN treats the queuing delay as an identical constant for each node, while in reality the queuing delay varies among nodes due to different traffic load of each node. Moreover, although the weekly flight trajectories and schedules are similar, there exist a certain mismatch between the historical flight data and the current flight data. Hence, there may exist errors between the minimum delay

[3]Although only considering the top-K ranked neighbors may potentially degrade the performance, when K is sufficiently large, the performance degradation is negligible as shown in our simulation results in Section V.



(a) Without feedback (b) With feedback

Fig. 4. Comparison of the available information at the forwarding node for determining the next hop.



(a) Compute $\tilde{D}_*^{[t]}(2,i_d)$ (b) Compute $\tilde{D}_*^{[t]}(3,i_d)$

(c) Compute $\tilde{D}_*^{[t]}(4,i_d)$

Fig. 5. Compute $\tilde{D}_*^{[t]}(j_1,i_d)$, $j_1 \in \mathcal{C}_{p_n}^{[t]}$ for the first round to determine the update order for decoupling the mutual dependence.

$D_*^{[t]}(j,i_d)$ and its estimate $\hat{D}_*^{[t]}(j,i_d)$, which degrades the online routing performance. In the following, we introduce a feedback mechanism for improving the online routing decisions.

5) On-line Decision with Feedback: Let $j_1 \in \mathcal{C}_{p_n}$ denote the index of the next-hop candidate of p_n and $j_2 \in \mathcal{C}_{j_1}$ denote the index of the next-hop candidates of j_1, i.e., the next-but-one or *next-2-hop* candidate of p_n. Instead of directly computing $\hat{D}_*^{[t]}(j_1,i_d)$ by inputting the geographic information $\boldsymbol{x}_{p_n}^{[t]}$ observed by the forwarding node p_n into the DNN, we let each next-hop candidate $j_1 \in \mathcal{C}_{p_n}$ compute $\hat{D}_*^{[t]}(j_2,i_d)$ by inputting their local geographic observation $\boldsymbol{x}_{j_1}^{[t]}$ into the DNN and measure the link delay $D^{[t]}(j_1,j_2)$. Then, by letting $j_1 \in \mathcal{C}_{p_n}$ report $D^{[t]}(j_1,j_2)$ and $\hat{D}_*^{[t]}(j_2,i_d)$ back to the forwarding node p_n, the minimum delay from j_1 to the destination can be estimated based on the optimal substructure (10) using $D^{[t]}(j_1,j_2)$ and $\hat{D}_*^{[t]}(j_2,i_d)$. In the sequel, we provide an example snapshot for illustrating the on-line routing decision process relying on the feedback mechanism.

In Fig. 4, we first compare the information for determining the next hop acquired both with and without the feedback mechanism in an example snapshot. Let us assume that the packet is currently located at node 1, i.e., $p_n = 1$, and the maximum number of neighbors to be considered as candidates is $K = 3$. As shown in Fig. 4(a), the next-hop candidates of node 1 are node 2, 3 and 4. The next-hop candidates of nodes 2, 3 and 4 are $\{5,6\}$, $\{4,7,8\}$ and $\{3,8,9\}$, respectively, as shown in Fig. 4(b).

Without the feedback mechanism, node 1 first observes its local geographic information $\boldsymbol{x}_1^{[t]}$, which is then used as the input of the DNN for estimating the minimum delays from nodes $\{2,3,4\}$ to the destination, i.e.,

$$[\hat{D}_*^{[t]}(2,i_d), \hat{D}_*^{[t]}(3,i_d), \hat{D}_*^{[t]}(4,i_d)] = \boldsymbol{f}(\boldsymbol{x}_1^{[t]};\boldsymbol{\theta}). \tag{21}$$

Meanwhile, node 1 measures the link delays from itself to nodes $\{2,3,4\}$, i.e., $[D^{[t]}(1,2), D^{[t]}(1,3), D^{[t]}(1,4)]$. According to (20), the next hop can be determined by

$$p^{n+1} = \arg\min_{j\in\{2,3,4\}} \left\{ D^{[t]}(1,j) + [1 - I_d(j)]\, \hat{D}_*^{[t]}(j,i_d) \right\}. \tag{22}$$

After introducing the feedback mechanism, the minimum delay from the next-hop candidates $\{2,3,4\}$ can be estimated in the following manner instead. Specifically, nodes $\{2,3,4\}$ first observe their local geographic information $\boldsymbol{x}_2^{[t]}$, $\boldsymbol{x}_3^{[t]}$ and $\boldsymbol{x}_4^{[t]}$, respectively. The minimum delays from all the next-2-hop candidates to the destination can be estimated based on the DNN by inputting $\boldsymbol{x}_2^{[t]}$, $\boldsymbol{x}_3^{[t]}$ and $\boldsymbol{x}_4^{[t]}$, respectively. For example, the minimum delays from nodes $\{5,6\}$ to the destination can be estimated by

$$\left[\hat{D}_*^{[t]}(5,i_d), \hat{D}_*^{[t]}(6,i_d)\right] = \boldsymbol{f}(\boldsymbol{x}_2^{[t]};\boldsymbol{\theta})[:2], \tag{23}$$

where $\boldsymbol{y}[:n]$ denotes the first n elements of vector $\boldsymbol{y}$. Similarly, we can obtain

$$\left[\hat{D}_*^{[t]}(4,i_d), \hat{D}_*^{[t]}(7,i_d), \hat{D}_*^{[t]}(8,i_d)\right] = \boldsymbol{f}(\boldsymbol{x}_3^{[t]};\boldsymbol{\theta}) \tag{24a}$$

$$\left[\hat{D}_*^{[t]}(3,i_d), \hat{D}_*^{[t]}(8,i_d), \hat{D}_*^{[t]}(9,i_d)\right] = \boldsymbol{f}(\boldsymbol{x}_4^{[t]};\boldsymbol{\theta}). \tag{24b}$$

Meanwhile, nodes 2, 3 and 4 measure the link delays $\{D^{[t]}(2,5), D^{[t]}(2,6)\}$, $\{D^{[t]}(3,4), D^{[t]}(3,7), D^{[t]}(3,8)\}$ and $\{D^{[t]}(4,3), D^{[t]}(4,8), D^{[t]}(4,9)\}$, respectively. Then, with the aid of the feedback mechanism, node 1 now becomes aware of the link delays between itself to its next-hop candidates as well as the link delays between the next-hop candidates and the next-2-hop candidates, as shown Fig. 4(b). This allows the forwarding node to select the next-hop more wisely, since more link information (i.e., the queuing delay of next-hop candidates as well as the propagation and transmission delay from each next-hop candidate to the corresponding next-2-hop candidates) can now be taken into consideration by recursively exploiting the optimal substructure.

Consequently, the minimum delay from nodes $\{2,3,4\}$ to the destination can be estimated using the optimal substructure instead of using (21), as illustrated in Fig. 5. This is formulated as follows

$$\tilde{D}_*^{[t]}(2,i_d) = \min_{j_2\in\{5,6\}} D^{[t]}(2,j_2) + \hat{D}_*^{[t]}(j_2,i_d) \tag{25}$$

$$\tilde{D}_*^{[t]}(3,i_d) = \min_{j_2\in\{4,7,8\}} D^{[t]}(3,j_2) + \hat{D}_*^{[t]}(j_2,i_d) \tag{26}$$

$$\tilde{D}_*^{[t]}(4,i_d) = \min_{j_2\in\{3,8,9\}} D^{[t]}(4,j_2) + \hat{D}_*^{[t]}(j_2,i_d). \tag{27}$$

It is worth noting that node 3 and node 4 are not only the next-hop candidates of the forwarding node 1, but they are also the next-hop candidates of each other. From (26) and (27), we can see that the computation of $\tilde{D}_*^{[t]}(3, i_\mathrm{d})$ and $\tilde{D}_*^{[t]}(4, i_\mathrm{d})$ depends on the estimate $\hat{D}_*^{[t]}(4, i_\mathrm{d})$ and $\hat{D}_*^{[t]}(3, i_\mathrm{d})$, respectively. This suggests that the estimate $\tilde{D}_*^{[t]}(4, i_\mathrm{d})$ can be computed by

$$\begin{aligned}\tilde{D}_*^{[t]}(4, i_\mathrm{d}) = \min \Big\{ & D^{[t]}(4,3) + \tilde{D}_*^{[t]}(3, i_\mathrm{d}), \\ & D^{[t]}(4,8) + \hat{D}_*^{[t]}(8, i_\mathrm{d}), \\ & D^{[t]}(4,9) + \hat{D}_*^{[t]}(9, i_\mathrm{d}) \Big\} \end{aligned} \tag{28}$$

instead, where $\tilde{D}_*^{[t]}(3, i_\mathrm{d})$ is computed by (26). Compared with using (27), the real-time link delays of node 3 to its next-hop candidates, i.e., $\{D(3,4), D(3,8), D(3,9)\}$, are also taken into account when computing $\tilde{D}_*^{[t]}(4, i_\mathrm{d})$ using (28) and (26). Alternatively, $\tilde{D}_*^{[t]}(3, i_\mathrm{d})$ can be also computed by

$$\begin{aligned}\tilde{D}_*^{[t]}(3, i_\mathrm{d}) = \min \Big\{ & D^{[t]}(3,4) + \tilde{D}_*^{[t]}(4, i_\mathrm{d}), \\ & D^{[t]}(3,8) + \hat{D}_*^{[t]}(8, i_\mathrm{d}), \\ & D^{[t]}(3,9) + \hat{D}_*^{[t]}(9, i_\mathrm{d}) \Big\} \end{aligned} \tag{29}$$

in a similar manner to (28). Consequently, the computation of $\tilde{D}_*^{[t]}(3, i_\mathrm{d})$ and $\tilde{D}_*^{[t]}(4, i_\mathrm{d})$ are mutually dependent on each other, as shown in (28) and (29).

The mutual dependence arises from the fact that the packet can be sent between nodes 3 and 4 in either direction. However, link $4 \to 3$ and link $3 \to 4$ cannot co-exist in the optimal path. In the following, we decouple the mutual dependence by specifying the direction of the link between the mutually dependent candidates and by specifying the order of mutually dependent candidates for the minimum delay estimation.

In general, we define

$$\mathcal{M}_{p_n}^{[t]} \triangleq \left\{ j_2 \,\middle|\, \left(j_2 \in \mathcal{C}_{p_n}^{[t]} \right) \wedge \left(\exists j_1 \in \mathcal{C}_{p_n}^{[t]} \ s.t.\ j_2 \in \mathcal{C}_{j_1}^{[t]} \right) \right\} \tag{30}$$

as the *mutual candidate set*, which contains the specific next-hop candidates of p_n that are also the next-2-hop candidates of p_n. Since the delay to the destination decreases along the optimal path, the packet should only be sent to a node having a lower delay to the destination than that of the forwarding node. Therefore, the nodes in $\mathcal{M}_{p_n}^{[t]}$ are sorted according to the estimated minimum delay to the destination in ascending order. Specifically, let m_k denote the kth element of $\mathcal{M}_{p_n}^{[t]}$. We have $\tilde{D}_*^{[t]}(m_1, i_\mathrm{d}) \le \tilde{D}_*^{[t]}(m_2, i_\mathrm{d}) \le \cdots \le \tilde{D}_*^{[t]}(m_{|\mathcal{M}_{p_n}^{[t]}|}, i_\mathrm{d})$.

For the example considered, assume that we have $\tilde{D}_*^{[t]}(3, i_\mathrm{d}) < \tilde{D}_*^{[t]}(4, i_\mathrm{d})$ based on the value computed from (26) and (27). Then, the link $3 \to 4$ is less likely to exist in the optimal path. Therefore, we recompute the value of $\tilde{D}_*^{[t]}(3, i_\mathrm{d})$ and $\tilde{D}_*^{[t]}(4, i_\mathrm{d})$ as follows

$$\tilde{D}_*^{[t]}(3, i_\mathrm{d}) = \min_{j_2 \in \{7,8\}} \left\{ D^{[t]}(3, j_2) + \hat{D}_*^{[t]}(j_2, i_\mathrm{d}) \right\} \tag{31}$$

$$\begin{aligned}\tilde{D}_*^{[t]}(4, i_\mathrm{d}) = \min \Big\{ & D^{[t]}(4,3) + \tilde{D}_*^{[t]}(3, i_\mathrm{d}), \\ & D^{[t]}(4,8) + \hat{D}_*^{[t]}(8, i_\mathrm{d}), \end{aligned}$$

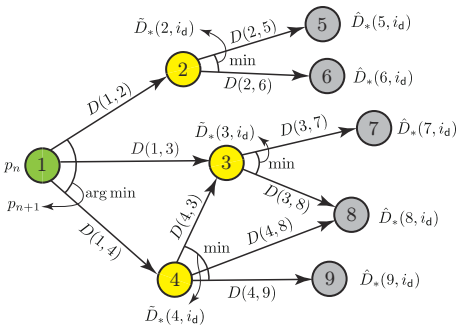


Fig. 6. The computation graph for determining the next hop p_{n+1}.

$$D^{[t]}(4,9) + \hat{D}_*^{[t]}(9, i_\mathrm{d}) \Big\}, \tag{32}$$

where node 4 is no longer considered as the next-hop candidate of node 3 and the computation of $\tilde{D}_*^{[t]}(4, i_\mathrm{d})$ depends on the value of $\tilde{D}_*^{[t]}(3, i_\mathrm{d})$. Finally the next hop can be determined by

$$p_{n+1} = \arg \min_{j_1 \in \{2,3,4\}} \left\{ D^{[t]}(1, j_1) + [1 - I_\mathrm{d}(j)]\, \tilde{D}_*^{[t]}(j_1, i_\mathrm{d}) \right\}. \tag{33}$$

The computation graph constructed for determining the next hop p_{n+1} is summarized in Fig. 6 after decoupling the mutual dependence.

In general, the computation of $\tilde{D}_*^{[t]}(j_1, i_\mathrm{d})$ for $j_1 \in \mathcal{C}_{p_n}$ is formulated as follows. We first initialize the value of $\tilde{D}_*^{[t]}(j_2, i_\mathrm{d})$ for each $j_2 \in \mathcal{C}_{j_1}$ and each $j_1 \in \mathcal{C}_{p_n}$ by $\tilde{D}_*^{[t]}(j_2, i_\mathrm{d}) \leftarrow \hat{D}_*^{[t]}(j_2, i_\mathrm{d})$.

Next, we compute the value of $\tilde{D}_*^{[t]}(j_1, i_\mathrm{d})$ for each $j_1 \in \mathcal{C}_{p_n}$ in the first round as

$$\begin{aligned}\tilde{D}_*^{[t]}(j_1, i_\mathrm{d}) \leftarrow [1 - I_\mathrm{d}(j_1)] \cdot \min_{j_2 \in \mathcal{C}_{j_1}^{[t]}} \Big\{ & D^{[t]}(j_1, j_2) \\ & + [1 - I_\mathrm{d}(j_2)]\, \hat{D}_*^{[t]}(j_2, i_\mathrm{d}) \Big\}. \end{aligned} \tag{34}$$

After that, we find the mutual candidate set $\mathcal{M}_{p_n}^{[t]}$ defined in (30) and sort its elements according to $\tilde{D}_*^{[t]}(m_1, i_\mathrm{d}) \le \tilde{D}_*^{[t]}(m_2, i_\mathrm{d}) \le \cdots \le \tilde{D}_*^{[t]}(m_{|\mathcal{M}_{p_n}^{[t]}|}, i_\mathrm{d})$ using the value computed by (34). Moreover, we define

$$\begin{aligned}\bar{\mathcal{C}}_{m_k}^{[t]} = \mathcal{C}_{m_k}^{[t]} - \Big\{ j \,\Big|\, & \left(j \in \mathcal{C}_{m_k}^{[t]} \right) \wedge \left(j \in \mathcal{M}_{p_n}^{[t]} \right) \\ & \wedge \left(\tilde{D}_*^{[t]}(j, i_\mathrm{d}) > \tilde{D}_*^{[t]}(m_k, i_\mathrm{d}) \right) \Big\}, \end{aligned} \tag{35}$$

which excludes the specific next-hop candidates of m_k that are also in the sorted mutual candidate sets $\mathcal{M}_{p_n}^{[t]}$ and have a higher minimum delay to the destination than that of m_k.

Then, we recompute the value of $\tilde{D}_*^{[t]}(j_1, i_\mathrm{d})$ for $j_1 \in \mathcal{M}_{p_n}^{[t]}$ one by one according the rank of j_1 in $\mathcal{M}_{p_n}^{[t]}$. To be more specific, for $k = 1, \cdots, \left|\mathcal{M}_{p_n}^{[t]}\right|$, we compute

$$\begin{aligned}\tilde{D}_*^{[t]}(m_k, i_\mathrm{d}) \leftarrow [1 - I_\mathrm{d}(m_k)] \cdot \min_{j_2 \in \bar{\mathcal{C}}_{m_k}^{[t]}} \Big\{ & D^{[t]}(m_k, j_2) \\ & + [1 - I_\mathrm{d}(j_2)]\, \tilde{D}_*^{[t]}(j_2, i_\mathrm{d}) \Big\}. \end{aligned} \tag{36}$$

Finally, the next hop is determined by the optimal substruc-

ture as follows

$$p_{n+1} = \arg\min_{j \in \mathcal{C}_{p_n}^{[t]}} \left\{ D^{[t]}(p_n, j) + [1 - I_{\mathsf{d}}(j)]\, \tilde{D}_*^{[t]}(j, i_{\mathsf{d}}) \right\}. \quad (37)$$

In Algorithm 1, we summarized procedure of the DL-aided routing during the online decision phase.

Algorithm 1 DL-aided routing for minimizing the E2E delay

% `Obtain the information for determining the next hop`

The forwarding node p_n discovers its next-hop candidates $\mathcal{C}_{p_n}$ and measures the delay from itself to each next-hop candidate $D^{[t]}(p_n, j_1)$, $j_1 \in \mathcal{C}_{p_n}$.

Each next-hop candidate $j_1 \in \mathcal{C}_{p_n}$ computes $\hat{D}_*^{[t]}(j_2, i_{\mathsf{d}})$ using the DNN, measures $D^{[t]}(j_1, j_2)$ for $j_2 \in \mathcal{C}_{j_1}$, and feeds all the information back to node p_n.

% `Computations at the forwarding node for determining the next hop`

Initialize $\tilde{D}_*^{[t]}(j_2, i_{\mathsf{d}}) \leftarrow \hat{D}_*^{[t]}(j_2, i_{\mathsf{d}})$ for each $j_2 \in \mathcal{C}_{j_1}$, $j_1 \in \mathcal{C}_{p_n}$.

Update $\tilde{D}_*^{[t]}(j_1, i_{\mathsf{d}})$ using (34) for each $j_1 \in \mathcal{C}_{p_n}^{[t]}$.

Find $\mathcal{M}_{p_n}^{[t]}$ defined by (30) and sort the element of $\mathcal{M}_{p_n}^{[t]}$ by $\tilde{D}_*^{[t]}(m_1, i_{\mathsf{d}}) \leq \tilde{D}_*^{[t]}(m_2, i_{\mathsf{d}}) \leq \cdots \leq \tilde{D}_*^{[t]}(m_{|\mathcal{M}_{p_n}^{[t]}|}, i_{\mathsf{d}})$.

for $k = 1, \cdots, \left|\mathcal{M}_{p_n}^{[t]}\right|$ **do**

  Find $\bar{\mathcal{C}}_{m_k}^{[t]}$ defined by (35).

  Update $\tilde{D}_*^{[t]}(m_k, i_{\mathsf{d}})$ using (36).

Transmit the packet to

$$p_{n+1} = \arg\min_{j \in \mathcal{C}_{p_n}^{[t]}} \left\{ D^{[t]}(p_n, j) + [1 - I_{\mathsf{d}}(j)]\, \tilde{D}_*^{[t]}(j, i_{\mathsf{d}}) \right\}$$

Set $n \leftarrow n + 1$ and repeat steps $1 \sim 9$ until the packet reaches the destination.

IV. Multi-Objective Routing

In this section, we extend our DL-aided routing to a challenging multi-objective scenario that considers both the E2E delay as well as the path capacity and path lifetime. We first introduce some basic concepts of MOO and formulate the multi-objective routing problem for simultaneously minimizing the delay, maximizing the capacity and maximizing the path lifetime. Then, we solve the problem using global information for obtaining the true Pareto front, and finally proposes an efficient DL-aided multi-objective routing algorithm that relies solely on local information for making online routing decisions in practical AANETs.

A. Problem Formulation

A standard MOO problem can be formulated as

$$\min_{\boldsymbol{x}} \ \boldsymbol{g}(\boldsymbol{x}) = [g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \cdots, g_M(\boldsymbol{x})] \quad (38)$$

$$s.t.\ \boldsymbol{x} \in \mathcal{X}, \quad (39)$$

where we have the following definitions.

Definition 1. *Pareto dominance*: Given a distinct pair of solutions $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, $\boldsymbol{x}_1$ is said to *dominate* $\boldsymbol{x}_2$, if and only if:

1) $g_m(\boldsymbol{x}_1) \leq g_m(\boldsymbol{x}_2)$ for any $m \in \{1, 2, \cdots, M\}$;
2) There exist an $m \in \{1, 2, \cdots, M\}$ such that $g_m(\boldsymbol{x}_1) < g_m(\boldsymbol{x}_2)$.

Accordingly, in the objective space, $\boldsymbol{g}(\boldsymbol{x}_1)$ is said to *dominate* $\boldsymbol{g}(\boldsymbol{x}_2)$.

Definition 2. *Pareto optimality*: A solution $\boldsymbol{x} \in \mathcal{X}$ is said to be *Pareto optimal* if and only if there does not exist another point $\boldsymbol{x}' \in \mathcal{X}$ that dominates $\boldsymbol{x}$. Furthermore, all the Pareto-optimal solutions mapped in the objective space constitutes the *Pareto front*.

Definition 3. *Weak Pareto optimality*: A solution $\boldsymbol{x} \in \mathcal{X}$ is said to be *weak Pareto optimal* if and only if there does not exist another point $\boldsymbol{x}' \in \mathcal{X}$ such that $g_m(\boldsymbol{x}') < g_m(\boldsymbol{x})$ for any $m \in \{1, 2, \cdots, M\}$.

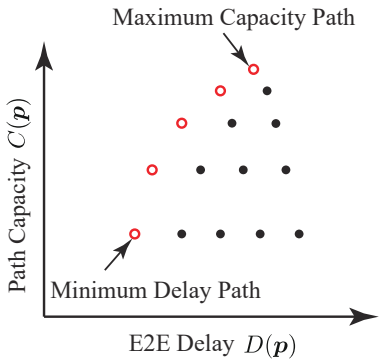


Fig. 7. Illustration of multi-objective routing.

Let us use a toy example to augment the above concepts more concretely. In Fig. 7, we consider a twin-objective routing problem, where the objectives are to simultaneously minimize the E2E delay and maximize the path capacity, i.e., $\min_{\boldsymbol{p}} \ [D(\boldsymbol{p}), -C(\boldsymbol{p})]$. Each point in Fig. 7 represents a possible path $\boldsymbol{p}$ from the source node to the destination node, while the x-coordinate and y-coordinate of a point indicate the delay and capacity of the path, respectively. Specifically, the black filled circles represent the non-Pareto-optimal solutions (paths), whose capacity can be increased or E2E delay can be reduced without sacrificing the other. By contrast, the red hollow circles represent the Pareto-optimal solutions (paths), which constitute the Pareto front. We can observe that the minimum delay and the maximum capacity cannot be achieved by a single path. Instead, there is a optimal trade-off relationship (characterized by the Pareto front) between the path capacity and E2E delay, i.e., which specifics the minimum increment of E2E delay required for achieving a certain path capacity improvement and vice versa.

Finally, the multi-objective routing problem minimizing the delay, maximizing the capacity and maximizing the path lifetime can be formulated in the following standard form

$$\mathsf{P}_{D,C,L}^{[t]}: \quad \min_{\boldsymbol{p}} \ [D^{[t]}(\boldsymbol{p}), -C^{[t]}(\boldsymbol{p}), -L^{[t]}(\boldsymbol{p})] \quad (40a)$$

$$s.t.\ (9b), (9c), (9d).$$

B. Multi-Objective Routing Using Global Information

In the following, we first solve problem $\mathsf{P}_{D,C,L}^{[t]}$ assuming global information is available. The ε-constraint method [16] has been developed for general MOO problems, which optimizes one of the metric in the OF and treats the remaining metrics as constraints. For the multi-objective routing problem $\mathsf{P}_{D,C,L}^{[t]}$, the following ε-constraint problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ can be formulated

$$\begin{aligned} \mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L): \quad \min_{\boldsymbol{p}} \ & D(\boldsymbol{p}) && (41a)\\ s.t.\ & C^{[t]}(\boldsymbol{p}) > \varepsilon_C && (41b)\\ & L^{[t]}(\boldsymbol{p}) > \varepsilon_L && (41c)\\ & (9b), (9c), (9d), \end{aligned}$$

where we consider E2E delay minimization as the objective while treat path capacity and lifetime as constraints because the resulting problem can be solved effortlessly by modifying a standard shortest path algorithm as shown later.

In general, a *weak Pareto-optimal* solution can be obtained by solving the ε-constraint problem [16]. Then, by varying the values of ε, multiple weak Pareto-optimal solutions can be found. In particular, according to [16, Theorem 4.2], we can obtain the following proposition.

Proposition 1. *If the optimal solution of the ε-constraint problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ is unique, it is also a Pareto-optimal solution of $\mathsf{P}_{D,C,L}^{[t]}$.*

As for the AANET routing problem, since the link delay $D^{[t]}(i,j)$ is a strictly increasing function of distance $d^{[t]}(i,j)$, which can be regarded as a random variable due to the uncertainty of flight position in 3D space, we can safely assume that $\Pr\big(D^{[t]}(\boldsymbol{p}_1) = D^{[t]}(\boldsymbol{p}_2)\big) = 0$ for arbitrary $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ that satisfies $\boldsymbol{p}_1 \neq \boldsymbol{p}_2$. Therefore, the optimal solution of $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ is unique. Then, by appropriately changing the value of $\varepsilon_C, \varepsilon_L$ and solving problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$, a series of Pareto-optimal solutions can be found.

Considering (6), the minimum capacity constraint (41b) is equivalent to $\min\limits_{n=1,\cdots,N-1} C^{[t]}(p_n,\ p_{n+1}) > \varepsilon_C$, which is further equivalent to $C^{[t]}(p_n, p_{n+1}) > \varepsilon_C, \forall n = 1, \cdots, N-1$. Similarly, considering (7), the minimum lifetime constraint (41c) is equivalent to $L^{[t]}(p_n, p_{n+1}) > \varepsilon_L, \forall n = 1, \cdots, N-1$. Therefore, problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ can be solved effortlessly by first deleting all the links that have a capacity no higher than ε_C or have a lifetime no longer than ε_L, and then applying any shortest path search algorithm. In Algorithm 2, we modified the Floyd-Warshall algorithm for solving problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ as an illustration.

In Algorithm 3, we show how to choose the value of ε_C and ε_L to find all the Pareto-optimal solutions of $\mathsf{P}_{D,C,L}^{[t]}$. Moreover, we have the following proposition regarding the Pareto optimality of Algorithm 3.

Proposition 2. *All the solutions found by Algorithm 3 are the Pareto-optimal solutions of $\mathsf{P}_{D,C,L}^{[t]}$ and all the Pareto-optimal solutions of $\mathsf{P}_{D,C,L}^{[t]}$ can be found by Algorithm 3.*

Algorithm 2 Modified the Floyd-Warshall algorithm for solving $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$

procedure COMPUTEMINIMUMDELAY
  Initialize $\mathsf{dist}[i,j] = \mathsf{inf}$, $\mathsf{next}[i,j] = \mathsf{null}$, $\forall i, j \in \mathcal{N}^{[t]}$
  for each link $i \to j$ **do**
    if $C^{[t]}(i,j) \geq \varepsilon_C$ and $L^{[t]}(i,j) \geq \varepsilon_L$ **then**
      $\mathsf{dist}[i,j] \leftarrow D^{[t]}(i,j)$
  for each node i **do**
    $\mathsf{dist}[i,i] \leftarrow 0$
  for $k \in \mathcal{N}^{[t]}$, $i \in \mathcal{N}^{[t]}$, and $j \in \mathcal{N}^{[t]}$ **do**
    if $\mathsf{dist}[i,j] > \mathsf{dist}[i,k] + \mathsf{dist}[k,j]$ **then**
      $\mathsf{dist}[i,j] \leftarrow \mathsf{dist}[i,k] + \mathsf{dist}[k,j]$
      $\mathsf{next}[i,j] \leftarrow \mathsf{next}[i,k]$
procedure FINDPATH(i_s, i_d)
  if $\mathsf{next}[i,j] = \mathsf{null}$ **then**
    return $p^* = \mathsf{null}$
  $p_1 = i_\mathsf{s}$, $n = 1$
  while $p_n \neq i_\mathsf{d}$ **do**
    $p_{n+1} \leftarrow \mathsf{next}[p_n, i_\mathsf{d}]$
    $n \leftarrow n + 1$
  return $\boldsymbol{p}^* = (p_1, p_2, \cdots, p_n)$

Algorithm 3 POMOR: Find all the Pareto-optimal solutions of $\mathsf{P}_{D,R,L}^{[t]}$

Initialize $\varepsilon_L \leftarrow 0$, Pareto set $\mathcal{P} \leftarrow \emptyset$
while $\varepsilon_L \neq \mathsf{null}$ **do**
  Initialize $\varepsilon_C \leftarrow 0$, $\mathcal{L} \leftarrow \emptyset$
  while $\varepsilon_C \neq \mathsf{null}$ **do**
    Obtain the optimal solution $\boldsymbol{p}_{i_\mathsf{s}}^*$ of problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ using any shortest path algorithm.
    if $\boldsymbol{p}_{i_\mathsf{s}}^* = \mathsf{null}$ **then**
      $\varepsilon_C \leftarrow \mathsf{null}$, $\varepsilon_L \leftarrow \mathsf{null}$
    else
      $\varepsilon_C = D^{[t]}(\boldsymbol{p}_{i_\mathsf{s}}^*)$
      Add $\boldsymbol{p}_{i_\mathsf{s}}^*$ and $L^{[t]}(\boldsymbol{p}_{i_\mathsf{s}}^*)$ into sets $\mathcal{P}$ and $\mathcal{L}$, respectively.
  $\varepsilon_L \leftarrow \min \mathcal{L}$
Output: The set of Pareto-optimal solutions $\mathcal{P}$

Proof: See Appendix. ∎

The complexity of Algorithm 3 is upper bounded by $\mathcal{O}(N_\mathsf{p}^2 N_\varepsilon)$, where N_p is the total number of Pareto-optimal solutions of problem $\mathsf{P}_{D,C,L}^{[t]}$and N_ε is the complexity of solving $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$. Specifically, when using the Floyd-Warshall algorithm, we have $N_\varepsilon = \mathcal{O}(|\mathcal{N}^{[t]}|^3)$. When using Dijkstra's algorithm, we have $N_\varepsilon = \mathcal{O}(N_\mathsf{e} + |\mathcal{N}^{[t]}| \log |\mathcal{N}^{[t]}|)$, where $N_\mathsf{e} \leq \left|\mathcal{N}^{[t]}\right|^2$ is the number of all possible links in the network.

C. DL-Aided Multi-Objective Routing Using Local Information

Again, global information is not available in practical AANETs. Hence, in the following, we propose the DL-

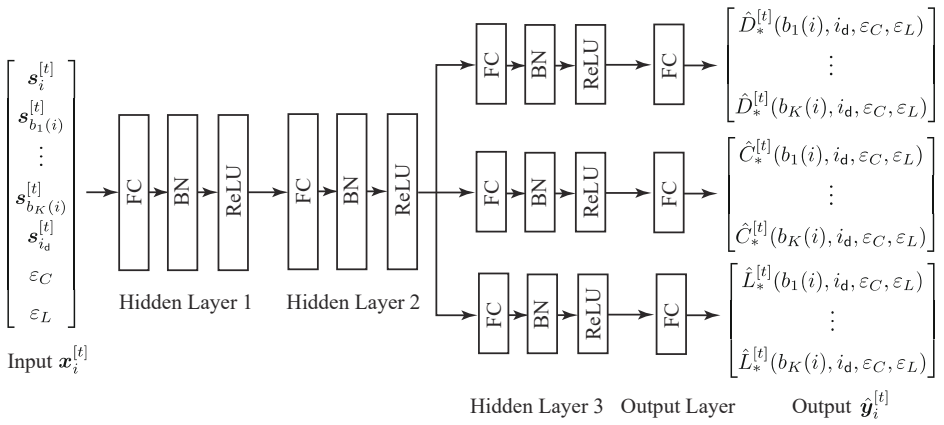


Fig. 8. DNN architecture for multi-objective routing.

aided multi-objective routing algorithm relying solely on local information.

Since the MOO problem $\mathsf{P}_{D,C,L}^{[t]}$ can be solved by solving a series of ε-constraint problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$, we propose the corresponding DL-aided routing algorithm for solving problem $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ relying solely on local information in the following. Similar to the minimum delay routing problem $\mathsf{P}_{D}^{[t]}$, $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ also has an optimal substructure, which can be formulated as

$$\begin{aligned} p_{n+1}^* = \ & \arg\min_{j\in\mathcal{B}_{p_n}^{[t]}} \Big\{ D^{[t]}(p_n, j) + D_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L) \\ & + I_{\mathsf{inf}}\Big[C^{[t]}(p_n, j) \le \varepsilon_C \text{ or } C_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L) \le \varepsilon_C\Big] \\ & + I_{\mathsf{inf}}\big[L^{[t]}(p_n, j) \le \varepsilon_L \text{ or } L_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L) \le \varepsilon_L\big]\Big\} \end{aligned} \tag{42}$$

where $I_{\mathsf{inf}}(X) = \mathsf{Inf}$ if X is true and $I_{\mathsf{inf}}(X) = 0$ otherwise, $D_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$, $C_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$ and $L_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$ are defined as follows. When there exists a path spanning from node j to node i_d satisfying the minimum capacity constraint ε_C and minimum lifetime constraint ε_L, $D_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$ and $C_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$ represent the delay, capacity, and lifetime achieved by the minimum-delay path satisfying the capacity and lifetime constraints. In particular, when there is no path from j to i_d that satisfies the capacity and lifetime constraint, we define $D_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$, $C_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$, and $L_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)$ as the delay, capacity, and lifetime achieved by the path from j to i_d whose capacity and lifetime are closest to ε_C and ε_L, respectively.

Since $[D^{[t]}(p_n, j), C^{[t]}(p_n, j), L^{[t]}(p_n, j)]$ can be readily measured by the forwarding node p_n, the optimal next hop can be determined locally based on (42), once the value of $[D_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L), C_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L), L_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)]$ of each next-hop candidate $j \in \mathcal{C}_i$ is available at p_n. Again, we use a DNN for estimating $\big[D_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L), C_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L), L_*^{[t]}(j, i_\mathsf{d}, \varepsilon_C, \varepsilon_L)\big]$. The structure of the DNN is shown in Fig. 8. Specifically, the input is

$$\boldsymbol{x}_{i,\varepsilon_C,\varepsilon_L}^{[t]} = \Big[\tilde{\boldsymbol{s}}_i^{[t]}, \tilde{\boldsymbol{s}}_{b_1(i)}^{[t]}, \cdots, \tilde{\boldsymbol{s}}_{b_K(i)}^{[t]}, \tilde{\boldsymbol{s}}_{i_\mathsf{d}}^{[t]}, \varepsilon_C, \varepsilon_L\Big] \in \mathbb{R}^{3(K+2)+2}, \tag{43}$$

where $\tilde{\boldsymbol{s}}_i^{[t]} = \big[\theta_i^{[t]}, \varphi_i^{[t]}, h_i^{[t]}, v_i^{[t]}, \delta_i^{[t]}\big]$. Compared with (11), the speed $v_i^{[t]}$ and heading $\delta_i^{[t]}$ of the airplane are further added into the feature, because the path lifetime depends on them. Moreover, ε_C and ε_L are also added into the feature to reflect the path capacity and lifetime requirements. The desired output of the DNN is

$$\begin{aligned} \boldsymbol{y}_{i,\varepsilon_C,\varepsilon_L}^{[t]} = \Big[& D_*^{[t]}(b_1(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \cdots, D_*^{[t]}(b_K(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \\ & C_*^{[t]}(b_1(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \cdots, C_*^{[t]}(b_K(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \\ & L_*^{[t]}(b_1(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \cdots, L_*^{[t]}(b_K(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \Big]. \end{aligned} \tag{44}$$

Again, we use $\boldsymbol{\theta}$ to denote the paremeters of the DNN. Then, the actual output of the DNN can be expressed as

$$\begin{aligned} \hat{\boldsymbol{y}}_{i,\varepsilon_C,\varepsilon_L}^{[t]} = & \ \boldsymbol{f}(\boldsymbol{x}_{i,\varepsilon_C,\varepsilon_L}^{[t]}; \boldsymbol{\theta}) \\ = & \Big[\hat{D}_*^{[t]}(b_1(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \cdots, \hat{D}_*^{[t]}(b_K(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \\ & \hat{C}_*^{[t]}(b_1(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \cdots, \hat{C}_*^{[t]}(b_K(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \\ & \hat{L}_*^{[t]}(b_1(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L), \cdots, \hat{L}_*^{[t]}(b_K(i), i_\mathsf{d}, \varepsilon_C, \varepsilon_L)\Big]. \end{aligned} \tag{45}$$

Algorithm 4 provides the details of generating training samples from historical flight data. The parameter $\boldsymbol{\theta}$ can be learned by minimizing the square error between the actual output and the desired output of the DNN, which is formulated as

$$\begin{aligned} & \min_{\boldsymbol{\theta}} \ \frac{1}{N_{\mathsf{total}}} \times \\ & \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}^{[t]}} \sum_{\varepsilon_C\in\mathcal{E}_C} \sum_{\varepsilon_L\in\mathcal{E}_L} \Big\| \boldsymbol{y}_{i,\varepsilon_C,\varepsilon_L}^{[t]} - \boldsymbol{f}(\boldsymbol{x}_{i,\varepsilon_C,\varepsilon_L}^{[t]}; \boldsymbol{\theta}) \Big\|^2. \end{aligned} \tag{46}$$

where $\mathcal{E}_C$ and $\mathcal{E}_L$ are sets of training samples for ε_C and ε_L, respectively.

Again, we can use Adam for training the DNN. After sufficient training, the DNN can be copied to each airplane for supporting the online routing decisions. Similar to Algorithm 1, the DL-aided routing algorithm conceived for solving $\mathsf{P}_{D}^{[t]}(\varepsilon_C, \varepsilon_L)$ can be obtained by exploiting the optimal substructure (42) instead, i.e., replacing (34) by (47),

$$\tilde{D}_*^{[t]}(j_1, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) = [1 - I_{\mathsf{d}}(j_1)] \min_{j_2 \in \mathcal{C}_{j_1}^{[t]}} \Big\{ D(j_1, j_2) + \lambda \left[\varepsilon_C - C^{[t]}(j_1, j_2)\right]^+ + \lambda \left[\varepsilon_L - L^{[t]}(j_1, j_2)\right]^+ + [1 - I_{\mathsf{d}}(j_2)] \cdot \Big[\hat{D}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) + \lambda \left[\varepsilon_C - \hat{C}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)\right]^+ + \lambda \left[\varepsilon_L - \hat{L}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)\right]^+ \Big] \Big\} \quad (47)$$

$$\tilde{D}_*^{[t]}(m_k, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_C) \leftarrow \min_{j_2 \in \tilde{\mathcal{C}}_{m_k}^{[t]}} \Big\{ D^{[t]}(m_k, j_2) + \lambda \left[\varepsilon_C - C^{[t]}(m_k, j_2)\right]^+ + \lambda \left[\varepsilon_L - L^{[t]}(m_k, j_2)\right]^+ + [1 - I_{\mathsf{d}}(j_2)]\, \tilde{D}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \Big\} \quad (48)$$

Algorithm 4 Generate training samples

Initialize $\mathcal{E}_L = \{\varepsilon_L^{(0)} + \alpha\Delta_L\}_{\alpha=1,\cdots,A}$ and $\mathcal{E}_C = \{\varepsilon_C^{(0)} + \beta\Delta_C\}_{\beta=1,\cdots,B}$
for $t \in \mathcal{T}_{\mathsf{train}}$ **do**
  $D'[i_{\mathsf{s}}] \leftarrow \mathsf{Inf}$, $C'[i_{\mathsf{s}}] \leftarrow 0$, $L'[i_{\mathsf{s}}] \leftarrow 0$ for all $i_{\mathsf{s}} \in \mathcal{N}^{[t]}$
  for $\alpha = 0, 1, \cdots, A$ **do**
    $D''[i_{\mathsf{s}}] \leftarrow D'[i_{\mathsf{s}}]$, $C''[i_{\mathsf{s}}] \leftarrow C'[i_{\mathsf{s}}]$, $L''[i_{\mathsf{s}}] \leftarrow L'[i_{\mathsf{s}}]$ for all $i_{\mathsf{s}} \in \mathcal{N}^{[t]}$.
    for $\beta = 0, 1 \cdots, B$ **do**
      Find the optimal solution $\boldsymbol{p}_{i_{\mathsf{s}}}^*$ of $\mathsf{P}_D^{[t]}(\varepsilon_L^{(0)} + \alpha\Delta_L, \varepsilon_C^{(0)} + \beta\Delta_C)$ for all $i_{\mathsf{s}} \in \mathcal{N}^{[t]}$ using Algorithm 2.
      for $i_{\mathsf{s}} \in \mathcal{N}^{[t]}$ **do**
        if $\boldsymbol{p}_{i_{\mathsf{s}}}^* \neq \mathsf{null}$ **then**
          $D_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leftarrow D(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$, $C_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}\varepsilon_C, \varepsilon_L) \leftarrow C(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$, $L_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leftarrow L(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$.
          $D''[i_{\mathsf{s}}] \leftarrow D(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$, $C''[i_{\mathsf{s}}] \leftarrow C(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$, $L''[i_{\mathsf{s}}] \leftarrow L(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$.
          if $\beta = 0$ **then**
            $D'[i_{\mathsf{s}}] \leftarrow D(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$, $C'[i_{\mathsf{s}}] \leftarrow C(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$, $L'[i_{\mathsf{s}}] \leftarrow L(\boldsymbol{p}_{i_{\mathsf{s}}}^*)$.
        else
          $D_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leftarrow D''[i_{\mathsf{s}}]$, $C_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leftarrow C''[i_{\mathsf{s}}]$, $L_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leftarrow L''[i_{\mathsf{s}}]$.
        if $D_*^{[t]}(i_{\mathsf{s}}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \neq \mathsf{Inf}$ **then**
          Save $\left(\boldsymbol{x}_{i_{\mathsf{s}}}^{[t]}, \boldsymbol{y}_{i_{\mathsf{s}}, \varepsilon_C, \varepsilon_L}^{[t]}\right)$ as a training sample.

replacing (36) by (48), and finally replacing (22) by

$$p_{n+1} = \arg \min_{j \in \mathcal{C}_{p_n}^{[t]}} \Big\{ D^{[t]}(p_n, j) + [1 - I_{\mathsf{d}}(j)]\, \tilde{D}_*^{[t]}(j, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) + \lambda \left[\varepsilon_L - L^{[t]}(p_n, j)\right]^+ + \lambda \left[\varepsilon_C - C^{[t]}(p_n, j)\right]^+ \Big\}, \quad (49)$$

where λ is the penalty coefficient introduced for penalizing the violation of the capacity or lifetime constraint. In practice, λ is set to a limited value instead of infinity used in (42). This is because when λ is excessive large, (47), (48) and (49) become extremely sensitive to the estimation error.

In Algorithm 5, we summarize the procedure of the DL-aided multi-objective routing during the online decision phase. Upon varying the values of ε_C as well as ε_L, and then using Algorithm 5 for the given ε_C and ε_L, multiple paths can be discovered as the solutions of the multi-objective routing problem $\mathsf{P}_{D,C,L}^{[t]}$.

Algorithm 5 DL-aided multi-objective routing with given ε_C and ε_L

% `Obtain the information for determining the next hop`
The forwarding node p_n discovers its next-hop candidates $\mathcal{C}_{p_n}$ and measures $\{D^{[t]}(p_n, j_1), C^{[t]}(p_n, j_1), L^{[t]}(p_n, j_1)\}$ for $j_1 \in \mathcal{C}_{p_n}$.
Each next-hop candidates $j_1 \in \mathcal{C}_{p_n}$ computes $\{\hat{D}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)$, $\hat{C}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)$, $\hat{L}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)\}$ using the DNN, measures $\{D^{[t]}(j_1, j_2), C^{[t]}(j_1, j_2), L^{[t]}(j_1, j_2)\}$ for $j_2 \in \mathcal{C}_{j_1}$, and feeds all the information back to p_n.
% `Computations at the forwarding node for determining the next hop`
Initialize $\tilde{D}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leftarrow \hat{D}_*^{[t]}(j_2, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)$ for each $j_1 \in \mathcal{C}_{p_n}, j_2 \in \mathcal{C}_{j_1}$.
Update $\tilde{D}_*^{[t]}(j_1, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)$ using (47) for each $j_1 \in \mathcal{C}_{p_n}$
Find $\mathcal{M}_{p_n}^{[t]}$ defined by (30) and sort the element of $\mathcal{M}_{p_n}^{[t]}$ by $\tilde{D}_*(m_1, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L) \leq \cdots \leq \tilde{D}_*(m_{|\mathcal{M}_{p_n}^{[t]}|}, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)$.
for $k = 1, \cdots, |\mathcal{M}_{p_n}^{[t]}|$ **do**
  Find $\tilde{\mathcal{C}}_{m_k}^{[t]}$ defined by (35).
  Update $\tilde{D}_*^{[t]}(m_k, i_{\mathsf{d}}, \varepsilon_C, \varepsilon_L)$ using (48).
Transmit the packet to p_{n+1} computed by (49).
Set $n \leftarrow n + 1$ and repeat steps $1 \sim 9$ until the packet reaches the destination.

V. Simulation Results

In this section, we evaluate and compare the performance of the proposed routing algorithms with benchmark algorithms via simulations based on real flight data.

A. Real Flight Data

To reflect different flight mobility patterns over different airspace, the flight data was collected over the North Atlantic ocean (i.e., NA scenarios) and the European continent (i.e.,

EU scenario) on two representative days. Specifically, the 25th of December typically has the quietest flight traffic and the 29th of June typically has the busiest flight traffic over a year. The status of each flight within the region considered was recorded in the format of [TS, latitude, longitude, altitude, speed, heading] for every 10 s over the complete 24 hours of each selected date.

In Fig. 9, we illustrate a snapshot of the flight data at 15:00 UTC on 25 Dec. 2017 for the NA scenario and on 25 Dec. 2018 for the EU scenario. The packet destination is chosen to be the London's Heathrow Airport (LHR) and Istanbul Airport (IST) for the NA scenario and EU scenario, respectively, as labeled in Fig. 10(a) and Fig. 10(b). For the NA scenario, it can be seen in Fig. 10(a) that strings of flights are heading towards the destination in a similar direction, while for the EU scenario shown in Fig. 9(b), the flight directions are quite heterogeneous.

B. Simulation Settings

Due to the limited real flight data collected, we generate more synthetic flight data based on the real flight data in order to train and test the DL-aided routing algorithms on different dataset. To reflect the mismatch between the training and testing environments, we randomly shift each flight's data along the timeline for generating the synthetic flight data for another day. By multiple realizations of the random shift based on the real flight data, multiple days of the synthetic flight data can be generated. Specially, the random shift is drawn from a Gaussian distribution with zero mean and a standard deviation of 30 min. We generate ten days of synthetic data based on the real flight data on Dec. 25 and Jun. 29, respectively, where six of them are used for training, three are used for validation (e.g., tuning the hyper-parameters), and one is used for testing outside the training and validation sets.

We further divided the training dataset into four time windows, each six hours, where a DNN was trained separately. In particular, for the simulations presented in the following, the DNN was trained using the training flight data within a time window of 12:00 $\sim$ 18:00 UTC. Once the DNN is well-trained, the DNN can be used for assisting the routing decisions during 12:00 $\sim$ 18:00 UTC in the testing dataset without the need of updating the DNN.

The transmit power is $P = 30$ dBm, the antenna gain is $G_{\mathsf{t}} = G_{\mathsf{r}} = 25$ dBi, and the carrier frequency is $f = 14$ GHz [22]. The transmission bandwidth is $W = 6$ MHz and the noise power is computed by $\sigma^2 = kTWF$, where $k = 1.3 \times 10^{-23}$, $T = 223.15$ Kelvin (i.e., -50 °C at an altitude of 10 km), $F = 4$ dB is the receiver's noise figure [23]. The packet size is $S = 1$ KBytes. To generate the labels for training the DNN, the queuing delay is set as a constant $D^{[t]}_{\mathsf{que}}(i) = 10$ ms for each node during training. By contrast, to reflect the heterogeneous traffic load of each node during testing, the queuing delay of each node is randomly drawn from a $[1, +\infty)$-truncated Gaussian distribution with a mean value of 10 ms and a standard deviation of 5 ms.

All experiments are conducted on an ordinary personal computer (PC) with AMD Ryzen™ 9 3950X CPU and a single Nvidia Geforce RTX™ 2080Ti GPU. The DNNs are implemented using TensorFlow 1.15 [24] on Windows 10.

C. Hyper-Paremeter Settings for DL-aided Routing Algorithms

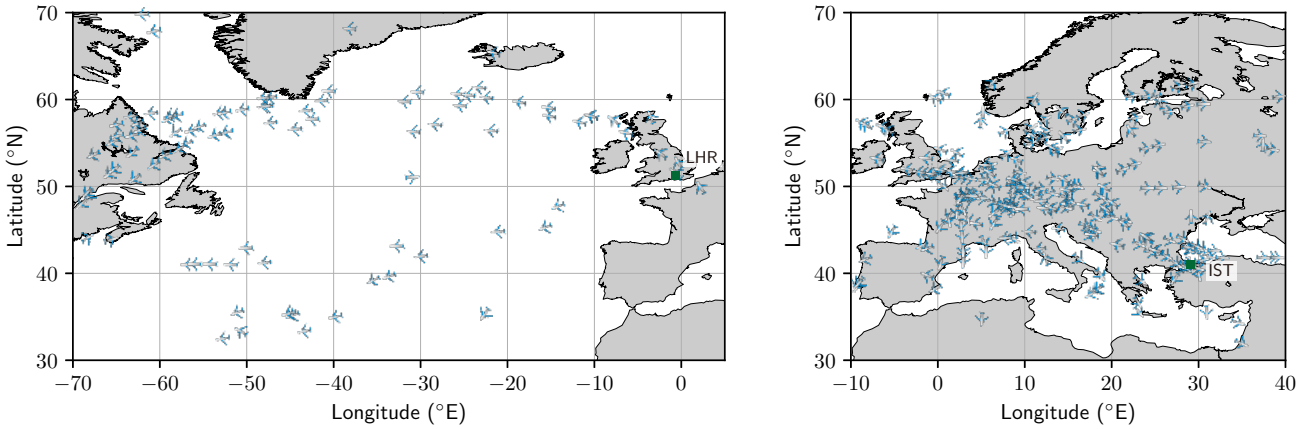
Each hidden layer of the DNN in Algorithm 1 has 100 neurons, as shown in Fig. 3. The first two hidden layers of the DNN in Algorithm 5 have 300 neurons and the third hidden layer has 100 neurons for each stream, as shown in Fig. 8. We use He's initialization [25] for all the hidden layers and all the output layers are initialized from the uniform distribution of $[-3 \times 10^{-3}, 3 \times 10^{-3}]$. The initial learning rate for the Adam optimizer is set to 10^{-3}. The mini-batch size for gradient descent is 1000. The maximal number of neighbors to be considered as the next-hop candidates is set to 10 for Algorithm 1 and set to 40 for Algorithm 5. For Algorithms 4 and 5, ε_C and ε_L are swept over $[20, 50]$ Mbps with a step size of 2 Mbps and $[0, 30]$ min with a step size of 5 min, respectively, in order to discover multiple paths. The penalty coefficient is set to $\lambda = 10$.

D. Performance Analysis and Comparison

The following routing protocols are considered for comparison:

1) *Optimal*: The optimal path found by solving the miminal delay problem $\mathsf{P}^{[t]}_D$ via the Floyd-Warshall algorithm, which relies on the global information regarding the delay of every possible link in the network.
2) *Greedy perimeter stateless routing (GPSR)*: The routing protocol proposed in [11], 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.
3) *Geographic Load Share Routing (GLSR)*: The routing protocol proposed in [12], which is based on the greedy forwarding used in GPSR and also takes the queuing delay of each next-hop candidate into consideration, when deciding about the next hop.
4) *Pareto Optimal*: The Pareto-optimal paths found by Algorithm 3, which relies on the global information regarding the delay, capacity and lifetime of every possible link in the network.

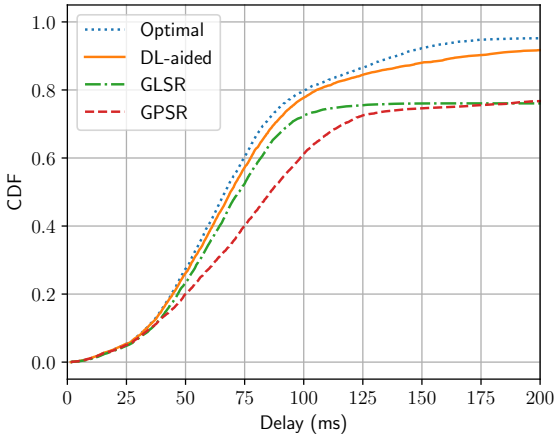
In Fig. 10, we compare the cumulative distribution function (CDF) curves of the E2E delay of all the flights on the quietest day, which can be regarded as the worst-case scenario for AANETs. In particular, we show the results during a time window of 15:00 $\sim$ 16:00 for a clear comparison. We can see that GLSR outperforms GPSR, because the queuing delay is taken into consideration. Our proposed DL-aided routing (Algorithm 1) performs close to the optimal policy, achieving lower E2E delay and higher success probability (defined as the probability of E2E delay lower than 200 ms) than GPSR and GLSR for both the NA and EU scenario. We note that



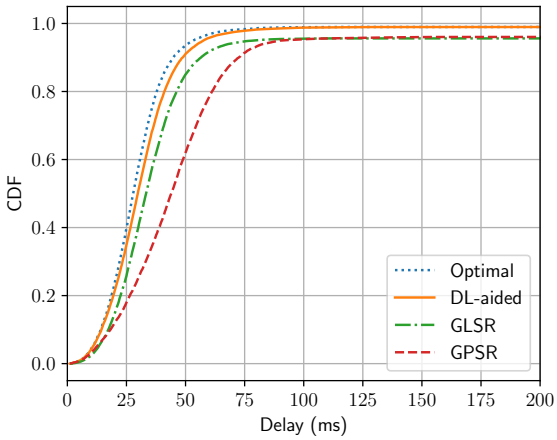
(a) NA scenario, 15:00 UTC time on 25 Dec. 2017.

(b) EU scenario, 15:00 UTC time on 25 Dec. 2018.

Fig. 9. Flight data over the North Atlantic ocean and the European continent.



(a) NA scenario, 15:00 ∼ 16:00 UTC Dec. 25, 2017.



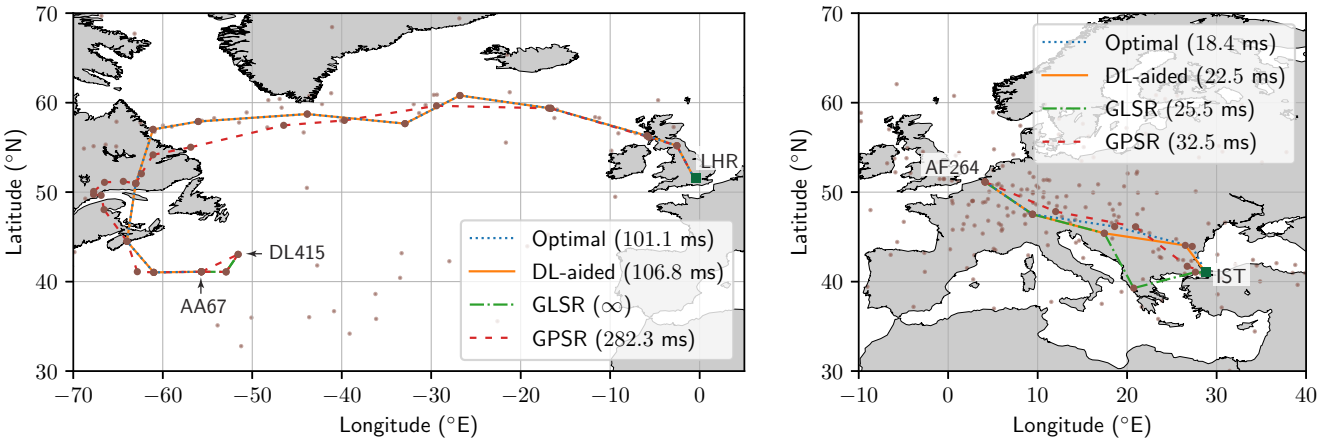
(b) EU scenario, 15:00 ∼ 16:00 UTC Dec. 25, 2018.

Fig. 10. E2E delay comparison for NA and EU scenarios.

similar conclusions can be made when testing on other time windows and on other days, which are not included here due to the space limitations.

To explain how the DL-aided routing algorithm outperforms the benchmark policies, we further compare the paths and their corresponding E2E delays found by different routing algorithms in an example snapshot for a particular flight in each scenario. In Fig. 11(a), we show the paths found for flight AA67 to LHR. We can see that GLSR encounters a communication void, when the packet is forwarded to flight DL415, and hence fails to find a path to the packet destination. By contrast, GPSR use perimeter routing after it encounters the communication void. Although GPSR can find a path to the destination eventually, it struggles through many hops to get around the void region. Since our DL-aided routing is trained using the historical flight data and hence is embedded with the topology knowledge, it can bypass the communication void more efficiently than GPSR, and find a similar path to the optimal path. In Fig. 11(b), we show the paths found for flight AF264 to IST. We can see that the flight distribution over the European continent is denser, and hence it is less likely to encounter a communication void. Consequently, all routing algorithms find paths with similar number of hops. Since GPSR does not consider the queuing delay, it achieves the highest E2E delay. In contrast, GLSR and our DL-aided routing algorithms achieve lower E2E delay. Benefited from the DNN and the feedback mechanism, each forwarding node can exploit more information using the DL-aided routing algorithm for determining the next hop, and hence DL-aided routing achieve lower E2E delay than GLSR and performs closely to the optimal path.

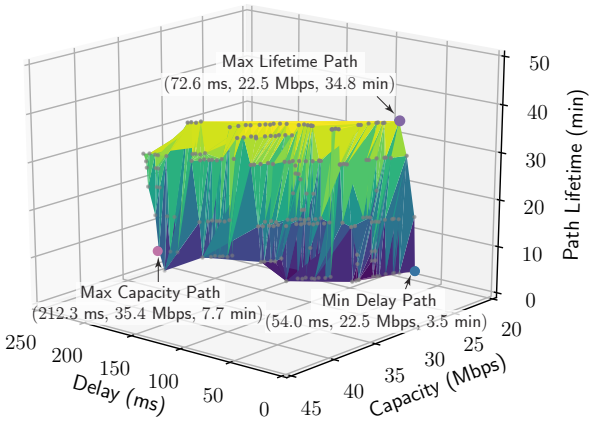
To test the routing algorithms when multi-component OF is considered, we use the flight data on the busiest day, where more paths are available between the source and destination nodes due to more airplanes in the sky. Specifically, we consider the paths from flight DL405 to LHR in the NA scenario and the paths from flight BA480 to IST in the EU scenario (both on 15:00 UTC Jun. 29, 2018) as examples for demonstrating the relationship between delay, capacity and lifetime achieved by the Pareto-optimal paths. In Fig. 12, we plot the Pareto front of the multi-objective routing problem $\mathrm{P}^{[t]}_{D,C,L}$ found by Algorithm 3, where each point represents a Pareto-optimal path and the surface interpolated by all the points visualizes the 3D Pareto front. Among all the Pareto-



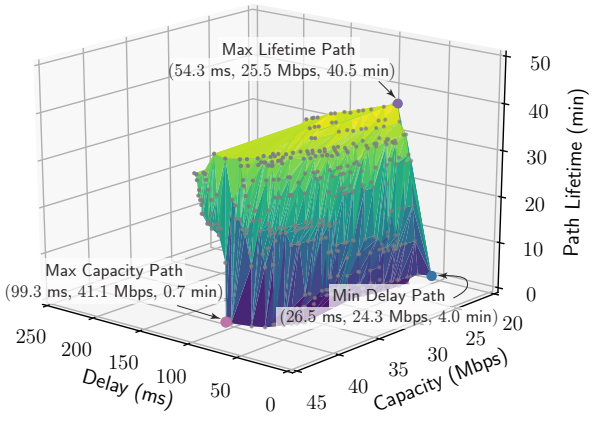
(a) NA scenario, 15:00 UTC Dec. 25, 2017.

(b) EU scenario, 15:00 UTC Dec. 25, 2018.

Fig. 11. Comparison of paths found by different routing algorithms.



(a) Flight DL405 to LHR, 15:00 UTC Jun. 29, 2018.



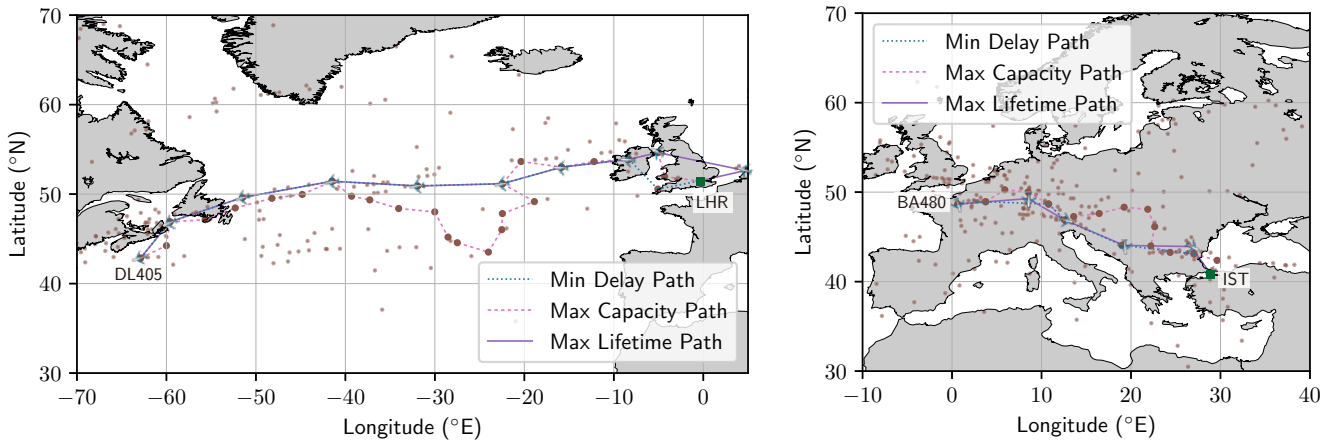
(b) Flight BA480 to IST, 15:00 UTC Jun. 29, 2018.

Fig. 12. 3D Pareto front.

optimal paths, we highlight the performance achieved by the minimum-delay path, the maximum-capacity path and the maximum-lifetime path. We can see that improving one metric on the Pareto front will sacrifice at least one of the other two metrics. For example, the minimum-delay path from flight DL405 to LHR has a delay of 54 ms, a capacity of 22 Mbps, and a lifetime of 3.5min. To increase the capacity to its maximum value of 35.4 Mbps, the lifetime increase to 7.7 min simultaneously while the delay increases to 212.3 ms. To increase the lifetime to its maximum value of 34.8 min, the capacity remains unchanged while the delay increases to 72.6 ms.

To further investigate the tradeoff relationship, we plot the minimum-delay path, maximum-capacity path, and maximum-lifetime path of the example snapshots in Fig. 13. We can see that to achieve the maximum capacity, the number hops increases to reduce the distance of each link, which results in longer total distance from the source to the destination and higher E2E delay. To show how the maximum-lifetime path is selected, we further plot the flight direction of the nodes on the maximum-lifetime path. We can see that, to achieve the maximum path lifetime, the angle between the flight directions of adjacent nodes is small so that the lifetime of each link is maximized. Moreover, from Fig. 13(a), we can see that most of the maximum-lifetime path share the same nodes with the minimum-delay path. This is because the flight directions are similar in the NA scenario, where strings of flights are heading toward the destination continent.

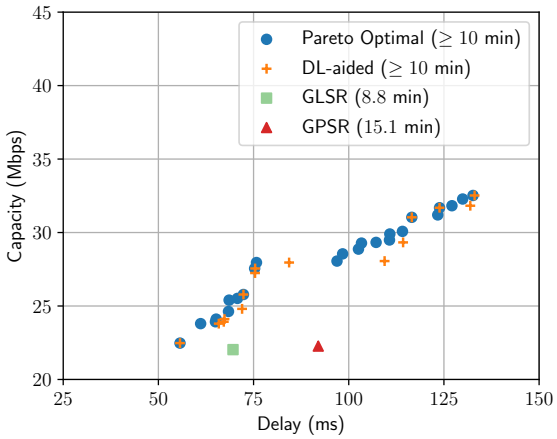
Since there exist many Pareto optimal paths as shown in Fig. 12 and maximizing the path lifetime may not be so important as minimizing the delay or maximizing the capacity once the path lifetime exceeds a certain threshold, in the following, we treat the lifetime as a constraint and consider the tradeoff between capacity and delay. Specifically, the minimum lifetime constraint is set to 10 min. We compare the capacity and delay of different routing algorithms in Fig. 14. Since GLSR and GPSR may not guarantee the minimum lifetime constraint, we also provide the corresponding path lifetime in the legend. We



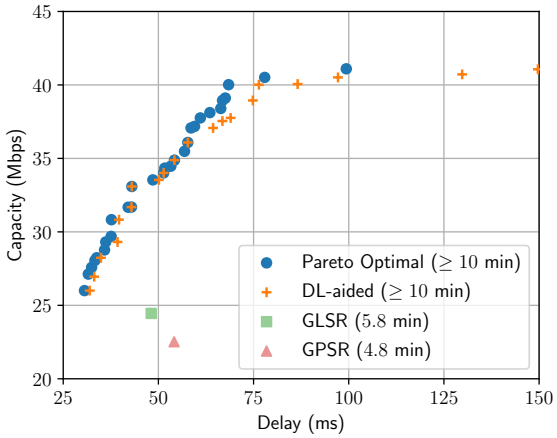
(a) Flight DL405 to LHR, 15:00 UTC Jun. 29, 2018.

(b) Flight BA480 to IST, 15:00 UTC Jun. 29, 2018.

Fig. 13. Minimum-delay path, maximum-capacity path, and maximum-lifetime path.



(a) Flight DL405 to LHR, 15:00 UTC Jun. 29, 2018.



(b) Flight BA480 to IST, 15:00 UTC Jun. 29, 2018.

Fig. 14. Capacity and delay tradeoff. The minimum lifetime constraint for "Pareto Optimal" and "DL-aided" is 10 min.

can see that our DL-aided routing (Algorithm 5) can find paths having a performance approaching to the Pareto front and dominate the paths found by GLSR and GPSR. Moreover, the lifetime constraint is guaranteed simultaneously. By contrast, GLSR fails to guarantee the lifetime constraint in both the NA and EU scenarios, while GPSR can only guarantee the lifetime constraint in the NA scenario, where most of the flight directions are similar.

TABLE III
COMPLEXITY COMPARISON

Complexity Indicator	SO-DNN	MO-DNN
Number of parameters to be trained	1.5×10^4	1.9×10^5
Input feature dimensions	36	212
Number of training samples	9.1×10^4	1.1×10^7
Training time per iteration	2.3 ms	7.9 ms
Number of iterations for training	2×10^3	10^4
Total training time	4.6 s	79 s
One-shot evaluation time	0.04 ms	0.09 ms

In Table III, we compare the training and testing complexity of the DNN used in single-objective routing and multi-objective routing, both in NA scenario on Jun. 29, 2018. For ease of expression, the DNN used for single-objective routing is termed as the SO-DNN, while the one used for multi-objective routing is termed as the MO-DNN. Since the MO-DNN has higher input dimension, more hidden layers and more neurons per layer for learning the relationship between the delay/capacity/lifetime and the thresholds, its number of parameters to be trained is about 13 times higher than that of SO-DNN. Moreover, because we have to sample different thresholds for the training of the MO-DNN, its training set is about a hundred times higher than that of the SO-DNN. As a result, the number of iterations required for convergence is 5 times higher compared to the training of SO-DNN. Further considering that the complexity per iteration of the MO-DNN's training is about 3.4 times higher than that of SO-DNN, the total training complexity of the MO-DNN is about 17 times higher than that of the SO-DNN, which can be done within minutes using an ordinary PC.

For the testing part, the multi-objective routing has to evaluate the MO-DNN's output for different input thresholds in order to discover multiple paths. Therefore, the testing complexity is proportional to the number of thresholds. Fortunately, we can combine the different thresholds into a batch and evaluate the MO-DNN's output for all the thresholds in parallel. Therefore, its computation time is similar to that in SO-DNN. Moreover, in the testing phase, only one-shot forward propagation is required for evaluating the DNN's output, which in general has significantly lower computational complexity than that of training relying on numerous iterations of back propagation. As a result, both the execution times for a one-shot evaluation for the outputs of SO-DNN and MO-DNN are less than 1 ms.

VI. Conclusions

In this paper, we proposed DL-aided routing policies for AANETs formed by passenger airplanes, where DNNs are invoked for learning the relationship between the local geographic information observed by the forwarding node and the information that is required for determining the optimal next hop. The DNN is trained by supervised learning based on historical flight data and it is then stored by each airplane for assisting their routing decisions during flight solely based on their local information in a distributed manner. We further extended the DL-aided routing algorithm to multi-objective scenarios, where the delay, capacity and path lifetime were jointly optimized. Our simulation results based on real flight data showed that the proposed DL-aided routing outperforms existing routing protocols in terms of their delay, capacity as well as path lifetime, and it is capable of approaching the Pareto front that is obtained using perfectly known global link 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. Moreover, MOO introduces extra the computational complexity compared to single-objective optimization, resulting in higher computation resource demand, as well as higher processing time and power. Future research may integrate satellites into the AANET for supporting truly global coverage and may investigate how to further reduce the complexity of MOO.

Appendix

According to Proposition 1 and bearing in mind that the solution of $\mathsf{P}_D^{[t]}(\varepsilon_C, \varepsilon_L)$ is unique, we can see that all the solutions found by Algorithm 3 are the Pareto optimal solutions of $\mathsf{P}_{D,C,L}^{[t]}$. In the following, we prove the rest of Proposition 2.

For notation simplicity, we omit the superscript "$[t]$" in what follows. Let $\boldsymbol{p}_{m,n}$ denote a feasible solution found by the mth iteration of the outer loop and the nth iteration of the inner loop of Algorithm 3, and let N_m denote the total number of the inner iterations of the mth outer iteration. Then, according to steps 9 and 11 of Algorithm 3, $\boldsymbol{p}_{m,n}$ is the optimal solution of $\mathsf{P}_D\big(C(\boldsymbol{p}_{m,n-1}), \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$, $n' \in \{1, \cdots, N_{m-1}\}$. Moreover, since $C(\boldsymbol{p}_{m,n}) > 0$, $\boldsymbol{p}_{m,n}$ is also a feasible solution of $\mathsf{P}_D\big(0, \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$. We first prove the following lemma.

Lemma 1. *There exists an $\tilde{n} \in \{1, \cdots N_m\}$ such that $[D(\boldsymbol{p}_{m,\tilde{n}}), -C(\boldsymbol{p}_{m,\tilde{n}})]$ dominates $[D(\boldsymbol{p}'), -C(\boldsymbol{p}')]$ for any feasible solution $\boldsymbol{p}'$ of $\mathsf{P}_D\big(0, \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$ that satisfies $\boldsymbol{p}' \notin \{\boldsymbol{p}_{m,n}\}_{n=1,\cdots,N_m}$.*

Proof: Considering step 9 of Algorithm 3 and bearing in mind that the solution of $\mathsf{P}_D^{[t]}(\varepsilon_C, \varepsilon_L)$ is unique, it is not hard to prove that $D(\boldsymbol{p}_{m,n}) < D(\boldsymbol{p}_{m,n+1})$ and $C(\boldsymbol{p}_{m,n}) < C(\boldsymbol{p}_{m,n+1})$. Assume that there exists another feasible solution of $\mathsf{P}_D\big(0, \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$, $\boldsymbol{p}' \notin \{\boldsymbol{p}_{m,n}\}_{n=1,\cdots,N_m}$ such that $[D(\boldsymbol{p}'), -C(\boldsymbol{p}')]$ is not dominated by $\boldsymbol{p}_{m,n}$ for any $n \in \{1, \cdots, N_m\}$. Further assume that there exists an $n_0 \in \{1, \cdots, N_m - 1\}$ such that $D(\boldsymbol{p}_{m,n_0}) < D(\boldsymbol{p}') < D(\boldsymbol{p}_{m,n_0+1})$. Since the optimal solution of $\mathsf{P}_D\big(\varepsilon_C, \varepsilon_L\big)$ is unique, $[D(\boldsymbol{p}_{m,n_0}), -C(\boldsymbol{p}_{m,n_0})]$ and $[D(\boldsymbol{p}_{m,n_0+1}), -C(\boldsymbol{p}_{m,n_0+1})]$ are non-dominated by $[D(\boldsymbol{p}), C(\boldsymbol{p})]$ for any feasible solution $\boldsymbol{p}$ of $\mathsf{P}_D\big(0, \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$ according to Proposition 1. Then, we can obtain $-C(\boldsymbol{p}_{m,n_0}) > -C(\boldsymbol{p}') > -C(\boldsymbol{p}_{m,n_0+1})$, i.e., $C(\boldsymbol{p}_{m,n_0}) < C(\boldsymbol{p}') < C(\boldsymbol{p}_{m,n_0+1})$. Considering that $\boldsymbol{p}_{m,n_0+1}$ is the optimal solution of $\mathsf{P}_D\big(C(\boldsymbol{p}_{m,n_0}), \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$ and $\boldsymbol{p}'$ is a feasible solution of $\mathsf{P}_D\big(C(\boldsymbol{p}_{m,n_0}), \min_{n'} L(\boldsymbol{p}_{m-1,n'})\big)$, we have $D(\boldsymbol{p}_{m,n_0+1}) \leq D(\boldsymbol{p}')$, which contradicts to $D(\boldsymbol{p}_{m,n_0}) < D(\boldsymbol{p}') < D(\boldsymbol{p}_{m,n_0+1})$. Similarly, we can readily obtain contradictions when $D(\boldsymbol{p}') < D(\boldsymbol{p}_{m,1})$, or when $D(\boldsymbol{p}') > D(\boldsymbol{p}_{m,N_m})$. Therefore, Lemma 1 is proved. ■

Next, we continue to prove the rest of Proposition 2. Assume that there exists another feasible solution of $\mathsf{P}_{D,C,L}$, i.e., $\boldsymbol{p}'' \notin \{\boldsymbol{p}_{m,n}\}$, that is also a Pareto-optimal solution of $\mathsf{P}_{D,C,L}^{[t]}$. According to step 11 of Algorithm 3, we have $\min_{n'} L(\boldsymbol{p}_{m-1,n'}) < \min_{n'} L(\boldsymbol{p}_{m,n'})$. Let M denote the total number of outer iterations and assume furthermore that there exist an $m_0 \in \{1, \cdots, M\}$ such that $\min_{n'} L(\boldsymbol{p}_{m_0-1,n'}) < L(\boldsymbol{p}'') \leq \min_{n'} L(\boldsymbol{p}_{m_0,n'})$. Since $C(\boldsymbol{p}'') > 0$ and $L(\boldsymbol{p}'') > \min_{n'} L(\boldsymbol{p}_{m_0-1,n'})$, $\boldsymbol{p}''$ is also a feasible solution of $\mathsf{P}(0, \min_{n'} L(\boldsymbol{p}_{m_0-1,n'}))$. Then, according to Lemma 1, there exists an $\tilde{n} \in \{1, \cdots, N_{m_0}\}$ such that $[D(\boldsymbol{p}_{m_0,\tilde{n}}), C(\boldsymbol{p}_{m_0,\tilde{n}})]$ dominates $[D(\boldsymbol{p}''), C(\boldsymbol{p}'')]$. Moreover, since $L(\boldsymbol{p}'') \leq \min_{n'} L(\boldsymbol{p}_{m_0,n'})$, we have $L(\boldsymbol{p}'') \leq L(\boldsymbol{p}_{m_0,\tilde{n}})$. Consequently, $[D(\boldsymbol{p}_{m_0,\tilde{n}}), C(\boldsymbol{p}_{m_0,\tilde{n}}), L(\boldsymbol{p}_{m_0,\tilde{n}})]$ dominates $[D(\boldsymbol{p}''), C(\boldsymbol{p}''), L(\boldsymbol{p}'')]$, which contradicts to the assumption that $\boldsymbol{p}''$ is Pareto-optimal. Similarly, we can easily obtain contradictions when $L(\boldsymbol{p}'') \leq \min_{n'} L(\boldsymbol{p}_{1,n'})$, or when $L(\boldsymbol{p}'') > \min_{n'} L(\boldsymbol{p}_{M,n'})$. Therefore, all the Pareto-optimal solutions of $\mathsf{P}_{D,C,L}^{[t]}$ can be found by Algorithm 3 and hence Proposition 2 is proved.

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] Q. Luo, J. Wang, and S. Liu, "Aeromrp: A multipath reliable transport protocol for aeronautical ad hoc networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3399–3410, Apr. 2018.

[4] A. Bujari, O. Gaggi, C. E. Palazzi, and D. Ronzani, "Would current ad-hoc routing protocols be adequate for the internet of vehicles? a comparative study," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3683–3691, Oct. 2018.

[5] 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.

[6] "Ad hoc on-demand distance vector (AODV) routing," IETF Netw. Working Group, Fremont, CA, USA, RFC 3561, Jul. 2003.

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

[8] 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.

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

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

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

[12] 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.

[13] 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.

[14] H. Yetgin, K. T. K. Cheung, and L. Hanzo, "Multi-objective routing optimization using evolutionary algorithms," in *Proc. IEEE WCNC*, 2012, pp. 3030–3034.

[15] 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.

[16] V. Chankong and Y. Y. Haimes, *Multiobjective decision making: theory and methodology*. New York: Elsevier Science, 1983.

[17] Z. Fei, B. Li, S. Yang, C. Xing, H. Chen, and L. Hanzo, "A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 550–586, First Quarter 2016.

[18] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1472–1514, Third Quarter 2020.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, Lille, France, 2015, pp. 448–456.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, Banff, Canada, 2014, pp. 1–15.

[22] M. Vondra, M. Ozger, D. Schupke, and C. Cavdar, "Integration of satellite and aerial communications for heterogeneous flying vehicles," *IEEE Netw.*, vol. 32, no. 5, pp. 62–69, Sept./Oct. 2018.

[23] 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.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE ICCV*, 2015, pp. 1026–1034.