Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A traffic-aware Q-network enhanced routing protocol based on GPSR for unmanned aerial vehicle ad-hoc networks*

Yi-ning CHEN[1], Ni-qi LYU[1], Guang-hua SONG[‡1], Bo-wei YANG[1], Xiao-hong JIANG[2]

[1]*School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China*

[2]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*

E-mail: ch19930611@zju.edu.cn; lvniqi@gmail.com; ghsong@zju.edu.cn; boweiy@zju.edu.cn; jiangxh@zju.edu.cn

Received Aug. 7, 2019; Revision accepted Dec. 8, 2019; Crosschecked Aug. 5, 2020

Abstract: In dense traffic unmanned aerial vehicle (UAV) ad-hoc networks, traffic congestion can cause increased delay and packet loss, which limit the performance of the networks; therefore, a traffic balancing strategy is required to control the traffic. In this study, we propose TQNGPSR, a traffic-aware Q-network enhanced geographic routing protocol based on greedy perimeter stateless routing (GPSR), for UAV ad-hoc networks. The protocol enforces a traffic balancing strategy using the congestion information of neighbors, and evaluates the quality of a wireless link by the Q-network algorithm, which is a reinforcement learning algorithm. Based on the evaluation of each wireless link, the protocol makes routing decisions in multiple available choices to reduce delay and decrease packet loss. We simulate the performance of TQNGPSR and compare it with AODV, OLSR, GPSR, and QNGPSR. Simulation results show that TQNGPSR obtains higher packet delivery ratios and lower end-to-end delays than GPSR and QNGPSR. In high node density scenarios, it also outperforms AODV and OLSR in terms of the packet delivery ratio, end-to-end delay, and throughput.

Key words: Traffic balancing; Reinforcement learning; Geographic routing; Q-network
https://doi.org/10.1631/FITEE.1900401 **CLC number:** TP183; TN919.72

1 Introduction

In a mobile ad-hoc network for unmanned aerial vehicles (UAVs), point-to-point communication is required for many applications, such as network relay and flight information collection (Bekmezci et al., 2013). Because of the movement of UAVs, the routing problem of finding an efficient path in an unstable situation with frequently changing topology is challenging for the network (Wu et al., 2012).

There are many routing protocols for ad-hoc networks. The optimized link state routing (OLSR) protocol, a proactive routing protocol, exchanges network topology information periodically and maintains the forwarding paths from one node to others. However, because of the frequently changing topology, it is difficult for the protocol to provide up-to-date routing information in the UAV ad-hoc network. Furthermore, it exchanges control messages frequently, bringing a considerable amount of control overhead to the network. Meanwhile, ad-hoc on-demand distance vector (AODV) routing, a reactive routing protocol, broadcasts the route request (RREQ) message only when data transmission is requested, thus reducing the control overhead of the network. It performs well in many cases, but brings significant delay in the route discovery stage. On-demand multipath routing (Kenta et al., 2006) is a multi-routing version of AODV. It can reduce the invocation of route discovery by multiple next-hops

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61501399) and the National Key R&D Program of China (No. 2018AAA0102302)

ORCID: Yi-ning CHEN, https://orcid.org/0000-0002-3435-2851; Guang-hua SONG, https://orcid.org/0000-0003-3330-4978

in a routing table. Nevertheless, the route discovery delay remains high.

To deal with the high control overhead in the network, several geographic routing protocols have been proposed, such as location aided routing (LAR) (Ko and Vaidya, 2000), distance routing effect algorithm for mobility (DREAM) (Basagni et al., 1998), and greedy perimeter stateless routing (GPSR) (Karp and Kung, 2000). GPSR is a typical geographic routing protocol. It uses two forwarding modes, i.e., greedy mode and perimeter mode, to reduce control overhead. It also reduces the route discovery delay. However, GPSR may increase the end-to-end delay when it uses perimeter mode to escape the void area. Furthermore, it suffers from traffic congestion in dense traffic networks.

In this study, we propose TQNGPSR, a traffic-aware Q-network enhanced routing protocol based on GPSR, for UAV ad-hoc networks. It is an extension of QNGPSR (Lyu et al., 2019), which is a Q-network enhanced routing protocol based on GPSR. Compared with QNGPSR, TGNGPSR enforces a traffic balancing strategy by considering the congestion information of neighbors. It also reduces the use of perimeter forwarding by evaluating the quality of a wireless link based on the neighbor information and the visiting information of packets. By employing the Q-network algorithm (Mnih et al., 2015), a reinforcement machine learning algorithm, the protocol compares the quality of the candidate links and chooses one with the maximal Q-value.

2 Background and related works

2.1 Traffic balancing

There are many routing protocols enhanced with various traffic balancing techniques. Software defined network (SDN) hybrid routing (Zhang et al., 2014) selects key source/destination node pairs based on the information from previous destination-based routing, and splits traffic into explicit routing for key pairs and destination-based routing for non-key pairs. This protocol needs explicit routing, which is quite difficult to implement in UAV ad-hoc networks.

Penalizing Exponential Flow-spliTting (PEFT) (Xu et al., 2011) is an enhanced version of open shortest path first (OSPF) with traffic balancing. It splits traffic among all paths and penalizes longer paths exponentially. However, PEFT needs to collect information about all paths to the destination node and maintain a larger routing table than OSPF. If one link in the routing table breaks, the routing table may become obsolete, which happens more frequently than OSPF.

The control traffic balancing protocol (Lin et al., 2016) is another routing protocol and focuses on the control flow of SDNs. It uses queueing network theory (Bolch et al., 2006), and regards the control traffic forwarding issue as a large-scale nonlinear optimization problem. A polynomial-time approximation algorithm (PTAA) is proposed to find an approximate and near optimal solution in the solution space. This protocol needs the global view of the network status, which relies on a centralized controller in SDNs and is unsuited for decentralized ad-hoc networks. In addition, it is designed for the control flow, which can not split data flow reasonably.

2.2 Reinforcement learning

Reinforcement learning is learning how to take an optimal action under a state. The process of reinforcement learning can be described as a Markov decision process (MDP) (Sutton and Barto, 1998). Q-learning (Watkins and Dayan, 1992) is a widely used form of reinforcement learning algorithm. It uses Q-table, an action-value table, to map state-action pairs to Q-values, and estimates the action-value function by the Bellman equation to update Q-values:

$$\begin{aligned} Q(s_i,a_i) =&(1-\alpha)Q(s_{i-1},a_{i-1}) \\ &+\alpha[R_i+\gamma\max_{a'\in A}Q(s_i,a')], \end{aligned} \tag{1}$$

where $Q(s_i,a_i)$ is the Q-value of state s_i and action a_i at the i^{th} iteration , R_i the reward if action a_i is taken at the i^{th} iteration, α the learning rate, and γ the discount factor. The agent uses a greedy strategy to take an action a with the maximal Q-value in A, and the maximal Q-value will be used to update the Q-value at the next iteration. As i approaches ∞, $Q(s_i,a_i)$ approaches the optimal function Q^*.

There are many advanced Q-learning algorithms. Q(λ)-learning (Peng and Williams, 1996) extends the one-step Q-learning algorithm to incremental multi-step Q-learning. The state-chain sequential feedback Q-learning algorithm (Ma et al., 2013) is presented to speed up the convergence of

Q-learning. A state chain is built during the search process, and after receiving the reward of an action, Q-values of these state-action pairs in the state-chain are sequentially updated.

Deep Q-network (DQN) (Mnih et al., 2015) is an algorithm designed to play computer games with raw inputs. It uses a convolutional neural network (CNN) called Q-network to approximate the action-value function. The Q-network is trained by adjusting the parameters to minimize the mean-squared error in the Bellman equation, which is represented by the loss function as

$$\begin{aligned} &L(\theta_i)\\ =&E\{[R_i+\gamma\max_{a'\in A}Q(s_i,a';\theta_i^-)-Q(s_i,a_i;\theta_i)]^2\}, \end{aligned} \tag{2}$$

where $R_i+\gamma\max_{a'\in A}Q(s_i,a';\theta_i^-)$ is the target, θ_i the parameter at iteration i, and θ_i^- the parameter at some iterations before i.

DQN uses the same greedy strategy as Q-learning to take an action and minimizes the loss function using the maximal estimated Q-value. After training process, the approximated function $Q(s_i,a_i;\theta_i)$ is used to estimate the optimal function Q^*.

2.3 Reinforcement learning based routing protocols

There are many routing protocols based on reinforcement learning. Q-routing (Boyan and Littman, 1994) uses Q-learning to evaluate the delay of each link and selects the link with the minimum Q-value. Dynamic dual-reinforcement-learning routing (Coutinho et al., 2015) evaluates the quality of experience (QoE) of each link by applying Q-learning in wireless mesh networks. It combines forward learning and backward learning to disseminate the information about the path towards the source node and destination node. PFQ-AODV (Wu et al., 2013) is a Q-learning enhanced routing protocol based on AODV. It evaluates each link in terms of bandwidth, mobility, and link quality, and uses fuzzy logic to adjust the learning rate of Q-learning.

QGrid (Li et al., 2014) is a Q-learning-based geographic routing protocol for vehicular ad-hoc networks (VANETs). It divides the region into several grids, and each grid has its own Q-value. The agent selects a grid with the maximal Q-value among the neighbor grids and selects the nearest node to the destination inside the selected grid. Since the Q-value is learned offline, it is limited for use in a network where the distribution of nodes is relatively stable. QGeo (Jung et al., 2017) is another geographic routing protocol based on Q-learning. It uses flexible discount factors to select a reliable link that has a high probability of connecting to a neighbor node. Since it needs to calculate the Q-value for each received node, the increased calculation overhead limits the performance when the network size becomes larger.

The Q-learning based congestion-aware (QCA) algorithm (Farahnakian e al., 2011) is a routing protocol designed for network-on-chip (NoC). The main idea of QCA is to consider not only the local but also the global traffic conditions when making routing decisions. Upon reception of a returning packet which contains local and global congestion information, it uses congestion information to evaluate the delay of each link as Q-value and selects links like Q-routing. QCA considers only 2D-mesh topology, which is not a suitable architecture for UAV ad-hoc networks. The changing topology makes the global congestion information obsolete.

In our previous work, we have proposed QNGPSR, a Q-network enhanced geographic routing protocol based on GPSR. This protocol improves GPSR using neighbor topology information and Q-network. QNGPSR has better performance than GPSR when overcoming the void area. However, it does not have a traffic balancing strategy to split traffic in a dense traffic network, and does not perform well on reducing the use of perimeter mode.

3 Traffic-aware Q-network enhanced geographic routing

Fig. 1 shows the mechanism of TQNGPSR. As shown in Fig. 1a, the main idea of TQNGPSR is to extract congestion information as a feature to estimate the quality of each link as the Q-value by Q-network. The routing decisions are determined by those estimated Q-values. As shown in Fig. 1b, it uses congestion information to adjust Q-values. Those Q-values are used as the targets to train the Q-network, so the estimated Q-values will be closer to the actual quality of links when facing congestion. Moreover, in TQNGPSR, the visit list of a packet is extracted as a feature to estimate the Q-value. It

will help nodes avoid the region that the packet has ever visited when making routing decisions, which reduces the use of perimeter mode.

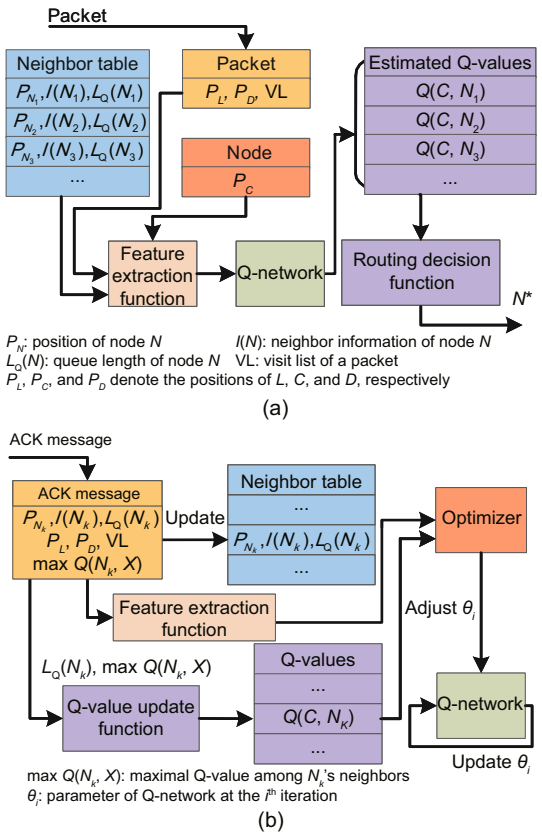

Fig. 1 Mechanism of TQNGPSR: (a) estimation process; (b) training process

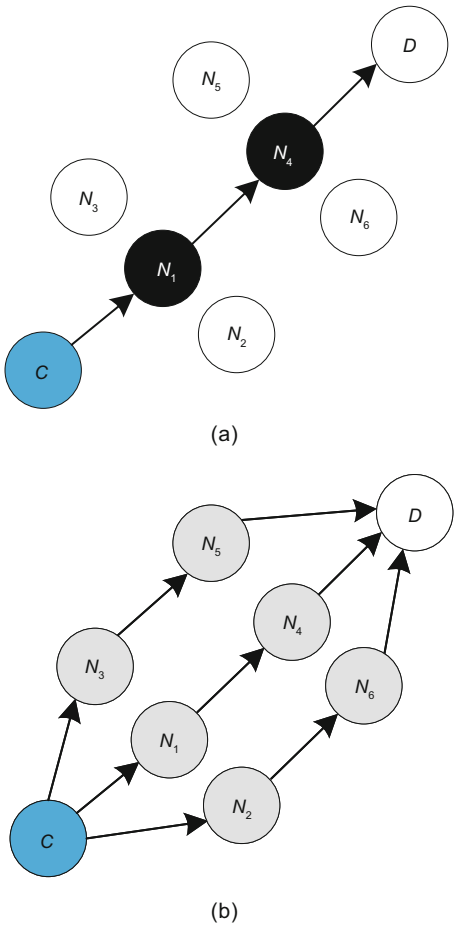

Fig. 2 Traffic balancing in dense traffic networks: (a) QNGPSR; (b) TQNGPSR

3.1 Traffic balancing

QNGPSR does not have a traffic balancing strategy, so it cannot solve traffic congestion in dense traffic networks. As shown in Fig. 2, the traffic is sent from the current node C to the destination node D. In Fig. 2a, node N_1 is located in a better position than N_2 and N_3, so C estimates N_1's Q-value as a large value; therefore, C selects N_1 as the forwarding node and sends a lot of packets to it. Since the forwarding capability of a node is limited, N_1 has to keep a lot of packets in its received buffer, which are waiting to be forwarded, thus increasing the end-to-end delay. Although QNGPSR uses the softmax policy to split some traffic among nodes whose Q-values are almost the same, the probability which is not correlated to the congestion information does not help with congestion. In addition, when its received buffer is full of packets, it has to drop other received packets, thus reducing the packet delivery ratio.

To solve this problem, TQNGPSR enforces a traffic balancing strategy. Unlike link-state routing protocols, TQNGPSR does not have the macroscopic view of the entire routing paths, so it needs to make routing decisions for each packet by analyzing the local topology and congestion information. In TQNGPSR, each node's congestion information is represented by the queue length in its received buffer, which is proportional to its queuing delay. When the current node C sends a data packet to the next-hop node N, N returns an acknowledge (ACK) message (containing its queue length) to C. Meanwhile, each node sends a HELLO message (also containing its queue length) to its neighbors periodically and updates its queue length table after receiving a message. Then, it estimates the Q-values without considering

congestion:

$$Q_{\mathrm{NC}}(C,N)=R+\gamma\max_{X\in\mathrm{NBR}(N)}Q(N,X), \tag{3}$$

where γ is the discount factor which is set to 0.9 based on our experience, $\mathrm{NBR}(N)$ the neighbor set of node N, and $\max_{X\in\mathrm{NBR}(N)}Q(N,X)$ the maximal Q-value among N's neighbors. The reward R is calculated as

$$R=\begin{cases}100, & \text{if } N\in\mathrm{NBR}(D),\\ -1, & \text{otherwise,}\end{cases} \tag{4}$$

where $\mathrm{NBR}(D)$ denotes the neighbor set of destination node D.

As shown in Fig. 3, node C receives a packet and sends the packet to destination node D. Nodes N_1, N_2, and N_3 are all neighbor nodes of C. N_1 is the nearest node to D with an almost full received buffer. N_3 is the farthest node to D with an empty received buffer, whose path to D has two hops. The distance between N_2 and D ($|N_2D|$) is between $|N_1D|$ and $|N_3D|$, and is with one hop to D. Some packets remain in N_2's received buffer. If congestion is not considered, because $|N_1D|<|N_2D|<|N_3D|$, then $Q(C,N_1)>Q(C,N_2)>Q(C,N_3)$. C selects N_1 to forward the packet. Since N_1's received buffer is almost full, N_1's queuing delay is considerable, so C should select another node to avoid congestion when making routing decisions. We reduce $Q(C,N_1)$ using a congestion penalty, which is calculated as a waiting reward (R_{W}) as follows:

$$R_{\mathrm{W}}(N)=\begin{cases}-\dfrac{L_{\mathrm{Q}}(N)}{L_{\mathrm{B}}}\beta_1 Q_{\mathrm{NC}}(C,N), & \\ \qquad\text{if } L_{\mathrm{Q}}(N)<0.9L_{\mathrm{B}}, & \\ -\beta_2 Q_{\mathrm{NC}}(C,N), \text{ otherwise,} & \end{cases} \tag{5}$$

where $L_{\mathrm{Q}}(N)$ is the queue length of N, L_{B} is the size of the received buffer, and β_1 and β_2 are different penalty factors in different congestion situations. $R_{\mathrm{W}}(N)$ is used for C to update the Q-value as

$$Q(C,N)=R_{\mathrm{W}}(N)+Q_{\mathrm{NC}}(C,N). \tag{6}$$

In some traffic balancing strategies, the congestion penalty is related only to congestion information. In Fig. 3, if the penalty factor β_1 is small, $Q(C,N_1)$ is not reduced sufficiently. C still selects N_1 and suffers from high queuing delay. If β_1 is large, $Q(C,N_2)$ is reduced too much. When N_2's queuing delay is tolerable, C selects N_3 and has to spend an extra hop, which brings extra delay. In TQNGPSR, $R_{\mathrm{W}}(N)$ is related to $L_{\mathrm{Q}}(N)$ and $Q_{\mathrm{NC}}(C,N)$, and β_1 is set to an appropriate value. Since $Q_{\mathrm{NC}}(C,N_1)$ is high and N_1's received buffer is almost full, the congestion penalty of N_1 is high. $Q(C,N_1)$ is reduced sufficiently and is less than $Q(C,N_2)$, and C is unlikely to select N_1. If N_2's queuing delay is tolerable, $Q(C,N_2)$ is reduced but is larger than $Q(C,N_3)$, and C selects N_2. If N_2's queuing delay is higher than the delay caused by an extra hop, C selects N_3 and spends an extra hop to reduce the delay.

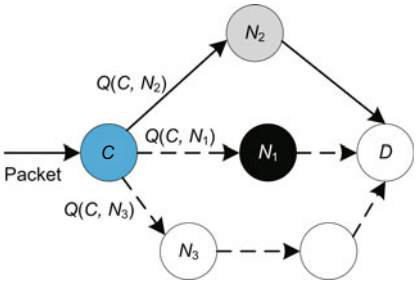

Fig. 3 Routing decision in congestion

However, congestion brings not only queuing delay but also packet loss when a node's received buffer is full, so we add an extra penalty to avoid the nodes with an almost full received buffer. In Eq. (5), β_2 is the penalty factor when $L_{\mathrm{Q}}(N)\geq 0.9L_{\mathrm{B}}$ and is larger than β_1.

TQNGPSR puts the queue length of neighbors into the feature extraction function to estimate Q-values, and adjusts parameters of the Q-network in the training stage. As shown in Fig. 2b, after the training stage, when N_1 is congested, the Q-value of N_1 decreases and C splits traffic to other nodes (such as N_2 and N_3). When N_1's congestion is alleviated, its Q-value increases and C will choose N_1 again. Therefore, the traffic can be balanced.

3.2 Q-network based route selection

In QNGPSR, when the current node is at the local minimum position and the greedy mode cannot forward packets, the current node switches to the perimeter mode to escape the void area. After the packet escapes from the local minimum position, the protocol switches back to the greedy mode. As shown in Fig. 4a, the current node C may have several neighbor nodes that have been visited by the packet, such as nodes L_1 and L_2. Since QNGPSR does not care about the positions of the neighbor

nodes that have forwarded the packet, C selects a node that has the maximal Q-value among its neighbor nodes as the next-hop node, such as node A. However, A may locate at a local minimum position and will still forward the packet by the perimeter mode, which may increase the number of hops.

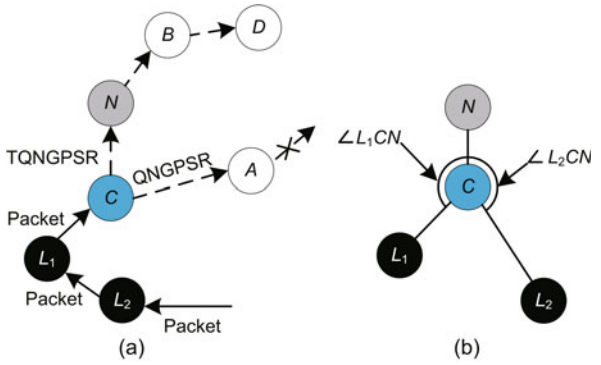

Fig. 4 $\angle L_iCN$ of the next-hop node, the current node, and the neighbor node which has been visited: (a) selection of current node C while forwarding a packet; (b) examples of $\angle L_iCN$

In TQNGPSR, when C receives a packet, it obtains the visit list (VL), which stores the node visited by the packet. As C knows that several neighbors have been visited, C can forward the packet to a node that is far from the area that the packet has visited, which can prevent the next-hop node from being located at a local minimum position. As shown in Fig. 4b, the minimum angle $\angle L_iCN$ represents the distance between the next-hop node and the visited area, where L_i is a neighbor of C in VL. C calculates the angle $\angle L_iCN$ as

$$\angle L_iCN = \arccos \frac{\overrightarrow{P_{L_i}P_C} \cdot \overrightarrow{P_NP_C}}{|P_{L_i}P_C||P_NP_C|}, \tag{7}$$

where P_{L_i}, P_C, and P_N denote the positions of nodes L_i, C, and N, respectively.

TQNGPSR puts neighbor nodes in VL to the feature extraction function to calculate the minimum $\angle L_iCN$. With the minimum angle, Q-network can estimate the Q-value of neighbors. As shown in Fig. 4a, TQNGPSR tends to forward packets to the node whose minimum angle is larger than those of other available nodes', such as N. This improvement avoids areas that the packet has already visited and decreases the use of perimeter mode, and thus the number of hops can be reduced.

We define the reinforcement learning algorithm in TQNGPSR as follows: Each packet is an agent and the node that holds the packet is considered the state of the agent, known as the current node C. Node C selects a candidate neighbor as the next-hop node N to forward the packet, which is considered the action of the agent. Unlike Q-learning based protocols, TQNGPSR does not maintain a Q-table to map Q-values to destination nodes. Instead, we use a Q-network to approximate the action-value function as $Q^*(C,N) \approx Q(C,N,\theta_i)$. Since different destination nodes have different features, the Q-values estimated by the trained Q-network should be different. Because it does not need to update Q-values for each destination node, DQN is suitable for a large-scale network.

As shown in Fig. 1a, after C receives a packet, it obtains a message containing VL of the packet, the position of the last-hop node L, and the destination node D. Meanwhile, C obtains information from its neighbor table, including the neighbors' position, neighbor information $I(N)$, and queue length $L_{\mathrm{Q}}(N)$. VL and the positions of L, C, and D are considered the features of the states. $I(N)$, $L_{\mathrm{Q}}(N)$, and the position of N are considered the features of actions. We replace the convolutional layers in DQN with a feature extraction function to extract and estimate the Q-values by the Q-network, where the feature extraction function is defined as

$$\begin{aligned}&\mathrm{feature}(P_L,P_C,P_D,\mathrm{VL},P_N,I(N),L_{\mathrm{Q}}(N))\\ =&[|P_NP_D|,|P_CP_D|,\min|P_{\mathrm{e}}(N,r)P_D|,\\ &\angle LCN,L_{\mathrm{Q}}(N),\min_{L_i\in\mathrm{VL}(C)}\angle L_iCN],\end{aligned} \tag{8}$$

where P_L, P_C, and P_D denote the positions of L, C, and D, respectively, VL the visit list of the packet, $\mathrm{VL}(C)$ the set of neighbors of C in VL (which means $\mathrm{VL}(C)=\mathrm{NBR}(C)\cap\mathrm{VL}$), P_N the position of N, $I(N)$ the neighbor information of N, and $L_{\mathrm{Q}}(N)$ the queue length of N. $\angle L_iCN$ is calculated using Eq. (7). $\angle LCN$ is the angle of L, C, and N, calculated as

$$\angle LCN = \arccos \frac{\overrightarrow{P_LP_C} \cdot \overrightarrow{P_NP_C}}{|P_LP_C||P_NP_C|}. \tag{9}$$

$|P_{\mathrm{e}}(N,r)P_D|$ in Eq. (8) denotes the distance between D and the estimated position of the neighbors of N in N's neighbor area r from $I(N)$. $P_{\mathrm{e}}(N,r)$ is defined as

$$\begin{aligned}&P_{\mathrm{e}}(N,r)\\ =&P_N+\begin{bmatrix}\cos\theta_{\mathrm{r}} & -\sin\theta_{\mathrm{r}}\\ \sin\theta_{\mathrm{r}} & \cos\theta_{\mathrm{r}}\end{bmatrix}\begin{bmatrix}\max_{X\in r}|P_NP_X|\\ 0\end{bmatrix},\end{aligned} \tag{10}$$

where θ_r represents the rotation angle of neighbor area r, which is $\frac{\pi}{8}+\frac{r-1}{4}\pi$. The neighbor information is defined in Lyu et al. (2019).

In the training stage, TQNGPSR uses a backward learning mechanism by the ACK message to disseminate and update Q-values. As shown in Fig. 5, after node N receives a data packet from node C, it sends an ACK message back to C, containing the maximal Q-value among its neighbors and the features like position, queue length, and neighbor information. As shown in Fig. 1b, after receiving an ACK message, the maximal Q-value and the queue length are used to calculate the target Q-value as $R_i(N)+\gamma Q(N, D; \theta_i^-)$, where θ_i^- is the parameter at the $(i-1)^{\text{th}}$ iteration. Using the target Q-value and the feature as the training sample, the Q-network is trained to minimize the loss function $L(\theta_i)$ through gradient descent.

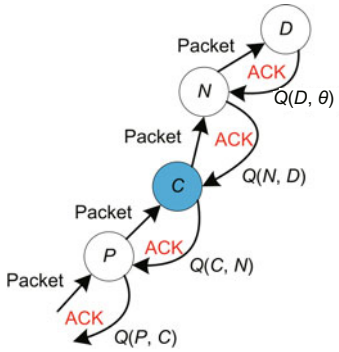

Fig. 5 Q-values disseminated by acknowledge (ACK) messages

After obtaining the estimated Q-value of each candidate neighbor, C takes an action with two methods, i.e., exploitation and exploration. As shown in Fig. 6, exploitation is to take the action based on the maximal estimated Q-value, and exploration is to take a random action. Using the ϵ-greedy policy with $\epsilon = 0.05$ (Mnih et al., 2015), C chooses exploitation with probability $1-\epsilon$ and exploration with probability ϵ.

In the running stage, parameters are constant, and nodes use the trained Q-network to estimate Q-values. We apply a softmax policy when making routing decisions as

$$P(a)=\frac{\exp\big(Q(a)/\tau\big)}{\sum_{a_i\in A}\exp\big(Q(a_i)/\tau\big)}, \tag{11}$$

where A is the set of agent actions and the parameter

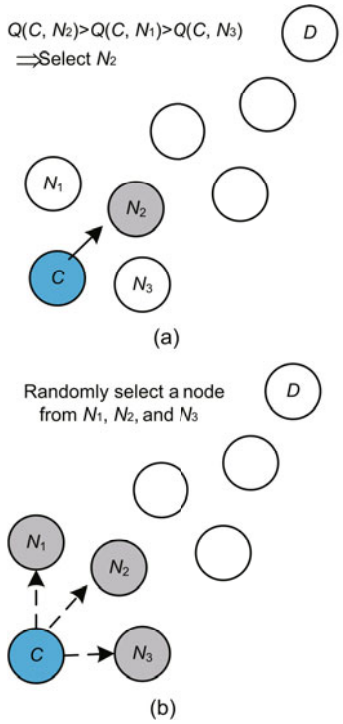

Fig. 6 Mechanisms of next-hop selection in the training stage: (a) exploitation; (b) exploration

τ is set to 0.05. The agent will take action a with probability $P(a)$, which splits traffic among nodes whose Q-values are almost the same.

4 Simulation results

To simulate the mobility model of micro-UAVs, we collect the flight trajectories around the Paris Charles de Gaulle Airport in November 2017 from https://www.flightradar24.com/ (Fig. 7). We rescale the movable region size and the velocity of flight to an appropriate size for micro-UAVs, such as ornithopters. The region size is 2000 m × 2000 m and the velocity of each node ranges from 3 to 10 m/s (Lyu et al., 2019).

We build a network simulator based on Python and SimPy to evaluate the performance of TQNGPSR and compare it with other protocols. We evaluate the performance of TQNGPSR when penalty factors β_1 and β_2 are set to different numbers. As summarized in Table 1, the number of nodes in test maps ranges from 60 to 100, and the transmission range is set to 350 m. Each node generates a HELLO message every 0.5 s. The size of received buffer of each node is 32 kb and its data transfer rate is 1 Mb/s. To simulate a dense traffic network, we set that each

node sends packets with 1 kb simulated data at various send rates, and that each node sends 1000 packets to a randomly selected destination node. When the node has sent all the 1000 packets, it will select another destination node randomly. We run the simulation 100 times in different maps, 100 s for each run. The warm-up stage takes 5 s and the remaining time is for the transmission stage.

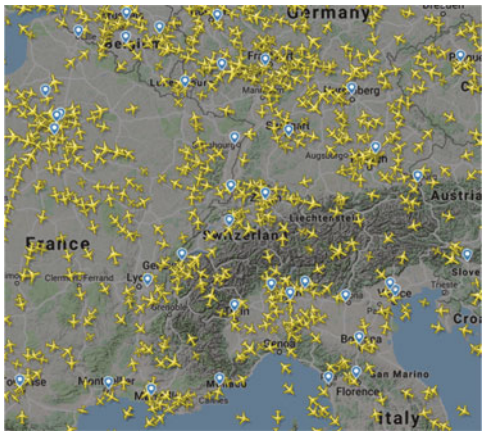

Fig. 7 Flight trajectories around the Paris Charles de Gaulle Airport

Table 1 Experiment parameters

Parameter	Setting
Movable region	2000 m×2000 m
Number of nodes	60–100
Node velocity	3–10 m/s
Transmission range	350 m
Packet data size	1 kb
Data transfer rate	1 Mb/s
HELLO message interval	0.5 s
Received buffer size	32 kb
Mobility model	Aircraft model
Simulation time	100 s

The Q-network in TQNGPSR is built in TensorFlow (Abadi et al., 2016). In the training stage, we deploy 90 nodes in the training maps and the nodes are static. We consider that the static node can make the Q-network approximate the optimal solution with a high convergence speed. The send rate in the training stage is set to 10 Hz. Other experimental parameters are the same as those in the test stage. We train TQNGPSR 100 times in different maps and the learning rate of the Q-network linearly decays by $0.01(1-e/100)$, where e is the number of training epochs.

4.1 Simulation results and comparison with other protocols

Fig. 8 shows the packet delivery ratio (PDR) of the routing protocols. AODV uses an on-demand route discovery strategy to keep nodes' routing table updated in time, and obtains higher PDR than TQNGPSR in a low node density network. However, as node density increases, the number of packets in the network also increases. Without the traffic balancing strategy, the nodes in AODV drop other received packets when their received buffers are full, so PDR of AODV does not increase with the increase of node density like in other routing protocols. OLSR, a proactive protocol, floods topology control (TC) messages to maintain the routing table. Since the destination nodes of packets are fixed in a short time in this scenario, the routing table may not be obsolete for packets. However, the same as AODV, OLSR does not perform better than TQNGPSR in a high node density scenario. As the number of nodes increases, OLSR cannot balance dense traffic, so it suffers from traffic congestion.

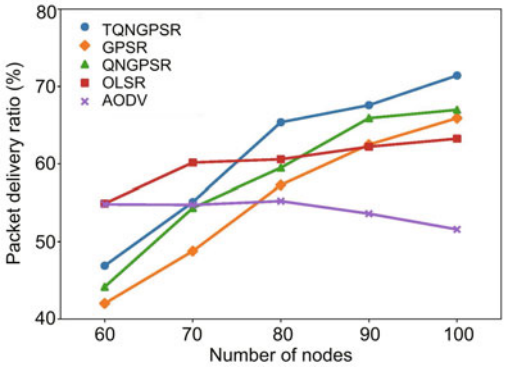

Fig. 8 Packet delivery ratio (send rate=10 Hz, β_1=0.1, β_2=0.5)

The performances of GPSR, QNGPSR, and TQNGPSR are not better than those of AODV and OLSR in a low node density scenario. However, as node density increases, GPSR, QNGPSR, and TQNGPSR have better performances. QNGPSR uses the Q-network to improve the performance of escaping void areas, so it obtains higher PDR than GPSR. However, there still has a lot of room for improvement. TQNGPSR outperforms GPSR and QNGPSR in various node density scenarios. TQNGPSR uses the neighbor information to avoid the area that has been visited by the packet, which reduces

the use of perimeter mode to escape void areas, and the probability of interruption can be decreased. Furthermore, TQNGPSR enforces the traffic balancing strategy. In a dense traffic network, GPSR and QNGPSR cannot solve traffic congestion, and packets will drop when the received buffers are full. In TQNGPSR, the traffic balancing strategy can balance received buffers, which makes packets free of packet loss.

Fig. 9 shows the end-to-end delay of the routing protocols. Fig. 10 shows the queuing delay of the routing protocols. AODV applies an on-demand route discovery strategy and discovers a new path before sending data packets. Moreover, since AODV does not care about traffic congestion, a large number of packets remain in the received buffers, which causes a huge queuing delay. OLSR obtains a shorter delay than AODV. Because of its nature, OLSR maintains the routing table proactively, so it spends less time in discovering a new path than AODV. However, as node density increases, the network becomes congested, so the queuing delay of OLSR becomes significant. In a high node density scenario, the delay of OLSR is higher than those of GPSR, QNGPSR, and TQNGPSR.

GPSR and QNGPSR provide a higher end-to-end delay than TQNGPSR. GPSR and QNGPSR does not perform so well when escaping void areas. The use of perimeter mode increases the number of hops, which brings a huge delay. Moreover, neither of them can balance the dense traffic, and the queuing delay also increases. TQNGPSR reduces the end-to-end delay in various node density scenarios. The improvement in escaping void areas can reduce the use of perimeter mode, which decreases the number of hops and the delay. Furthermore, the traffic balancing strategy can balance network congestion, and the queuing delay can be reduced.

Fig. 11 shows the number of hops of the routing protocols. Because of the above improvement, TQNGPSR outperforms GPSR and QNGPSR in terms of the number of hops. However, it obtains larger numbers than AODV and OLSR. Unlike AODV and OLSR, TQNGPSR does not maintain the routing table. TQNGPSR uses the neighbor position and queue length to estimate Q-values and tries its best to escape the void area. There are still some cases where the Q-values are incorrectly predicted. Since TQNGPSR must use perimeter

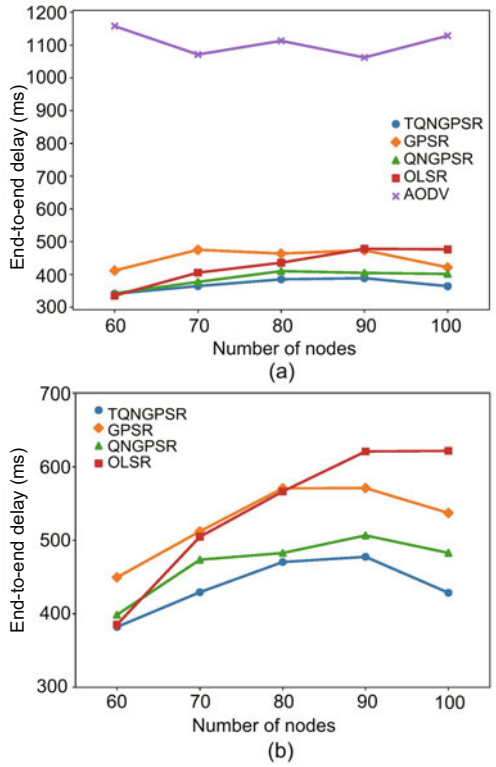

Fig. 9 End-to-end delay with (a) and without (b) AODV (send rate=10 Hz, β_1=0.1, β_2=0.5)

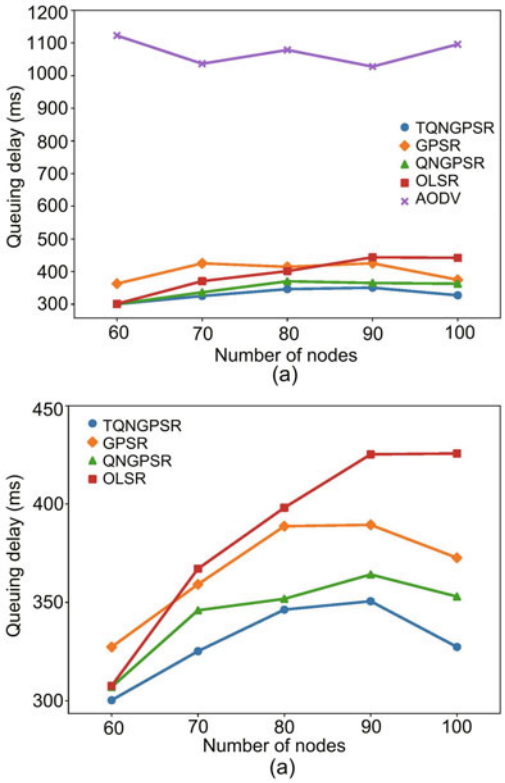

Fig. 10 Queuing delay with (a) and without (b) AODV (send rate=10 Hz, β_1=0.1, β_2=0.5)

forwarding in bad cases, the path may not be shorter than those of AODV and OLSR. Furthermore, to avoid nodes with a nearly full received buffer, TQNGPSR has to spend extra hops to reduce the queuing delay. Therefore, the end-to-end delay of TQNGPSR is lower than those of AODV and OLSR, though it obtains a large number of hops.

As shown in Fig. 11, the number of hops of TQNGPSR is reduced as the number of nodes increases. However, in Fig. 9b, as the number of nodes increases from 60 to 90, its end-to-end delay increases. When the number of nodes continues to increase to 100, its delay decreases. In a low node density scenario, the probability of facing a void area is high, and the use of perimeter mode is more frequent than that in a high node density scenario. Nodes in key positions are more likely to become congested and the received packets may drop. Because we count only the delay of the packets which are successfully received by the destination node, the dropped packets with a high queuing delay are not counted. As the number of nodes increases to 80, TQNGPSR can find more paths to avoid congestion at critical nodes. The end-to-end delay increases because some packets with a high queuing delay successfully reach the destination node. When the number of nodes increases to 100, TQNGPSR can balance dense traffic more efficiently, and the queuing delay and end-to-end delay are reduced.

Fig. 12a shows the control overhead of the routing protocols in different node density networks when the send rate is 10 Hz. AODV obtains the highest overhead among all protocols. Because of the frequent interruptions, AODV needs to send a large number of REER messages to the source node. The source node also needs to broadcast the RREQ messages frequently in the route discovery stage, which brings a large amount of overhead to the network. OLSR does not perform well. Since OLSR floods TC messages, the overhead increases linearly with the number of nodes. GPSR has only the HELLO message, and its performance in terms of control overhead is better than those of other protocols. The overhead of TQNGPSR is lower than those of AODV and OLSR, but a little higher than those of GPSR and QNGPSR. The size of the HELLO message of GPSR is only 12 bits (containing the position information). In TQNGPSR, however, the neighbor information (16 bits) and the queuing length (1 bit)

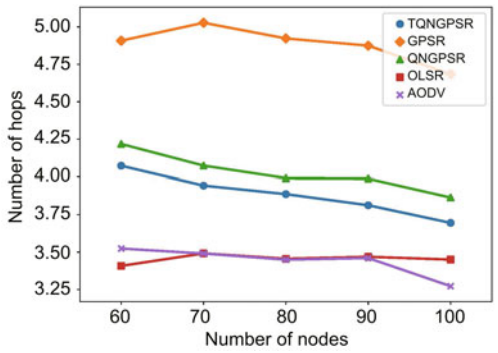

Fig. 11 Number of hops (send rate=10 Hz, β_1=0.1, β_2=0.5)

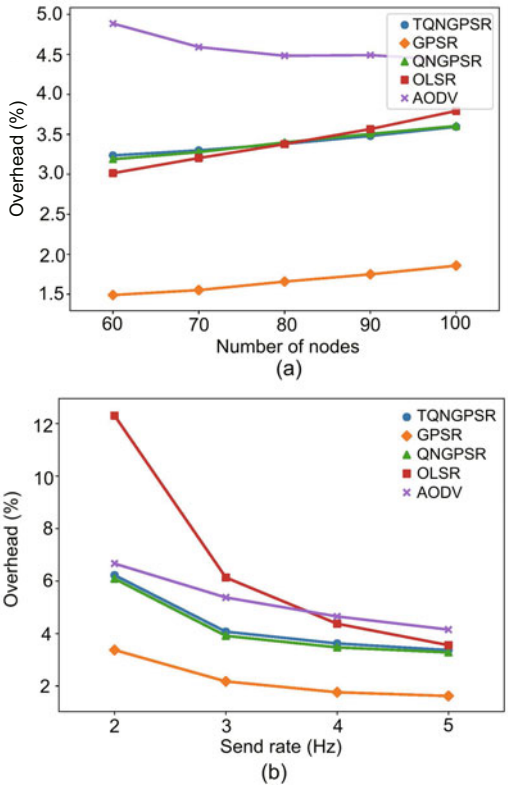

Fig. 12 Control overhead with different numbers of nodes (send rate=10 Hz) (a) and with different send rates (number of nodes=100) (b) ($\beta_1 = 0.1$ and $\beta_2 = 0.5$)

are contained in HELLO messages and acknowledge messages.

Fig. 12b shows the control overhead of the routing protocols at different send rates when the number of nodes is 100. Since the protocol header of GPSR is smaller than those of other protocols, GPSR provides the lowest overhead at various send rates. When the send rate is low, the number of TC messages in OLSR is relatively large, so the overhead of OLSR is significantly high. As the send rate increases, however,

the use of route discovery in AODV also increases, so AODV shows higher overhead in a network with a large send rate. The performance of TQNGPSR is better than those of AODV and OLSR, but worse than those of GPSR and QNGPSR.

Fig. 13 shows the average throughput of the routing protocols in different node density networks when the send rate is 10 Hz. The number shown in Fig. 13 represents the average number of pieces of data successfully received by each node per second in the transmission stage. In a low node density scenario, because of the nature of GPSR-like protocols, TQNGPSR does not perform better than AODV and OLSR. However, as node density increases, TQNGPSR outperforms other routing protocols. Because of the traffic balancing strategy, it can split dense traffic and solve the congestion while others cannot.

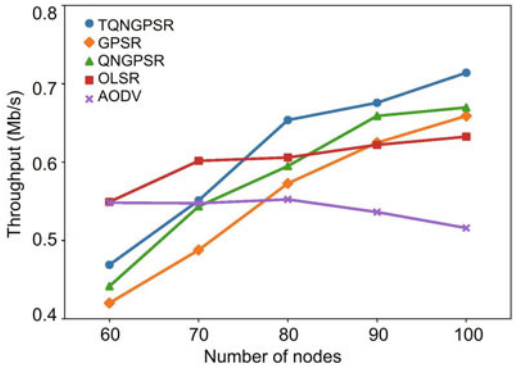

Fig. 13 Average throughput (send rate=10 Hz, $\beta_1 = 0.1$, $\beta_2 = 0.5$)

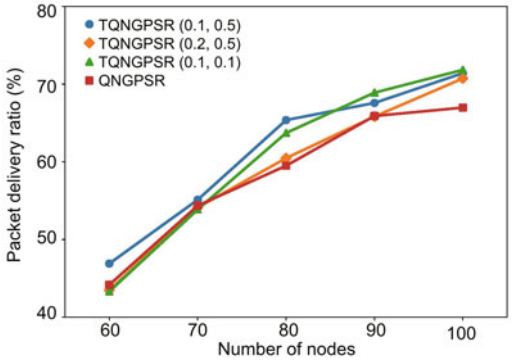

Fig. 14 Packet delivery ratio (send rate=10 Hz)

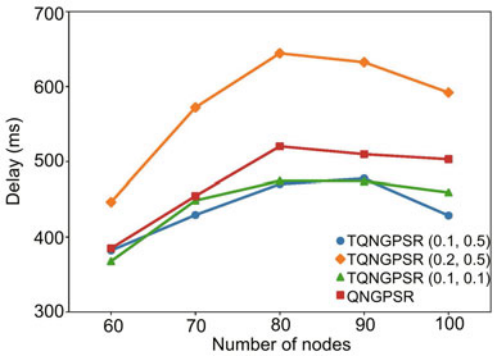

Fig. 15 End-to-end delay (send rate=10 Hz)

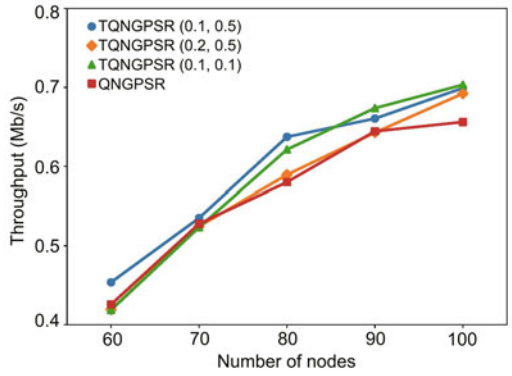

Fig. 16 Average throughput (send rate=10 Hz)

4.2 Simulation results under different penalty factors

We further simulate the performance of TQNGPSR under different penalty factors. Figs. 14, 15, and 16 show the PDR, end-to-end delay, and average throughout of TQNGPSR, respectively, when penalty factors (β_1, β_2) are set to (0.1, 0.5), (0.2, 0.5), and (0.1, 0.1).

TQNGPSR has good performance in terms of PDR, delay, and throughput when $\beta_1 = 0.1$, regardless of $\beta_2 = 0.1$ or 0.5. When β_1 increases to 0.2, the congestion penalty increases. The node in TQNGPSR would rather endure the cost of extra hops than select a node with a tolerable queue delay. Since the number of hops increases, the delay increases a lot when $\beta_1 = 0.2$. Moreover, the extra hops may increase the probability that packets cannot be forwarded, so the throughput and PDR decrease.

4.3 Complexity analysis and comparison with other protocols

In the simulations, a routing table entry in OLSR and AODV occupies six bits and contains the address of a destination node, the address of the next-hop, and the distance to the destination node. In OLSR, all the nodes in the network are stored in the routing table; thus, the storage complexity for a node is $O(n)$, where n is the number of nodes in the network. In AODV, only the destination nodes of the communication pairs are stored in the routing

table; thus, the storage complexity for a node is $O(n_c)$, where n_c is the number of communication pairs (Zhan and Zhou, 2008). A node in GPSR does not need the routing table; instead, it needs a neighbor table to store the positions and addresses of the neighbors (Karp and Kung, 2000). Thus, as in QNGPSR and TQNGPSR, the storage complexity of a node in GPSR is $O(m)$, where m is the number of neighbor nodes. Compared with GPSR, QNGPSR and TQNGPSR require extra space for a node to store the topology and congestion information for neighbors in the neighbor table. In the simulations, the size of each entry in the neighbor table of TQNGPSR is 27 bits. Though the size of an entry in the neighbor table is larger than that in the routing table, the number of entries of a neighbor table is much samller than that of a routing table. In the simulations, a neighbor table in TQNGPSR accupies less space than a routing table in OLSR or AODV.

The node in AODV spends round-trip time to discover a route. In the route discovery stage, the temporal complexity of AODV is $O(2h)$, where h is the maximum number of hops in each route (Shi and Deng, 2008). The temporal complexity of the establishment of a routing table in OLSR is $O(h)$, since the node needs h steps to send its topology information to the destination node. GPSR-like protocols do not spend time on route discovery. In the forwarding stage, the node in AODV or OLSR uses the routing table to forward packets, so the temporal complexity of each forward is $O(1)$. The node in GPSR-like protocols needs to evaluate its neighbors when forwarding packets. The temporal complexity of each forward is $O(m)$, where m is the number of neighbor nodes. QNGPSR and TQNGPSR use Q-network to evaluate neighbors, thus spending more time than GPSR, since GPSR needs only to calculate the distance between the destination node and each neighbor.

5 Conclusions

In this study, we have proposed TQNGPSR, a traffic-aware Q-network enhanced routing protocol based on GPSR, for the UAV ad-hoc network, which aims to improve the network performance when the traffic is dense. Simulation results showed that compared with GPSR and QNGPSR, TQNGPSR increases the packet delivery ratio and the average throughput, and reduces the end-to-end delay at the expense of a little higher control overhead. When the node number in the network increases, TQNGPSR performs better than AODV and OLSR in terms of the packet delivery ratio, end-to-end delay, and average throughput. In future work, we will try to lower the control overhead and improve the network performance of TQNGPSR by optimizing the feature extraction function as well as the neural network.

Contributors

Yi-ning CHEN and Guang-hua SONG designed the research. Yi-ning CHEN, Ni-qi LYU, and Xiao-hong JIANG processed the data. Yi-ning CHEN drafted the manuscript. Guang-hua SONG and Bo-wei YANG helped organize the manuscript. Yi-ning CHEN and Guang-hua SONG revised and finalized the paper.

Compliance with ethics guidelines

Yi-ning CHEN, Ni-qi LYU, Guang-hua SONG, Bo-wei YANG, and Xiao-hong JIANG declare that they have no conflict of interest.

References

Abadi M, Barham P, Chen JM, et al., 2016. TensorFlow: a system for large-scale machine learning. Proc 12th USENIX Conf on Operating Systems Design and Implementation, p.265-283.

Basagni S, Chlamtac I, Syrotiuk VR, et al., 1998. A distance routing effect algorithm for mobility (DREAM). Proc 4th Annual ACM/IEEE Int Conf on Mobile Computing and Networking, p.76-84. https://doi.org/10.1145/288235.288254

Bekmezci I, Sahingoz OK, Temel Ş, 2013. Flying ad-hoc networks (FANETs): a survey. *Ad Hoc Netw*, 11(3):1254-1270. https://doi.org/10.1016/j.adhoc.2012.12.004

Bolch G, Greiner S, de Meer H, et al., 2006. Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications (2nd Ed.). John Wiley & Sons, New York, USA.

Boyan JA, Littman ML, 1994. Packet routing in dynamically changing networks: a reinforcement learning approach. Proc 7th Int Conf on Neural Information Processing Systems, p.671-678.

Coutinho N, Matos R, Marques C, et al., 2015. Dynamic dual-reinforcement-learning routing strategies for quality of experience-aware wireless mesh networking. *Comput Netw*, 88:269-285. https://doi.org/10.1016/j.comnet.2015.06.016

Farahnakian F, Ebrahimi M, Daneshtalab M, et al., 2011. Q-learning based congestion-aware routing algorithm for on-chip network. Proc 2nd Int Conf on Networked Embedded Systems for Enterprise Applications, p.1-7. https://doi.org/10.1109/NESEA.2011.6144949

Jung WS, Yim J, Ko YB, 2017. QGeo: Q-learning-based geographic ad hoc routing protocol for unmanned robotic

networks. *IEEE Commun Lett*, 21(10):2258-2261. https://doi.org/10.1109/LCOMM.2017.2656879

Karp B, Kung HT, 2000. GPSR: greedy perimeter stateless routing for wireless networks. Proc 6th Annual Int Conf on Mobile Computing and Networking, p.243-254. https://doi.org/10.1145/345910.345953

Kenta T, Takeshi M, Shinya K, et al., 2006. Experimental evaluation of an on-demand multipath routing protocol for video transmission in mobile ad hoc networks. *J Zhejiang Univ-Sci A*, 7(S1):145-150. https://doi.org/10.1631/jzus.2006.AS0145

Ko YB, Vaidya NH, 2000. Location-aided routing (LAR) in mobile ad hoc networks. *Wirel Netw*, 6(4):307-321. https://doi.org/10.1023/A:1019106118419

Li RL, Li F, Li X, et al., 2014. QGrid: Q-learning based routing protocol for vehicular ad hoc networks. Proc 33rd Int Performance Computing and Communications Conf, p.1-8. https://doi.org/10.1109/PCCC.2014.7017079

Lin SC, Wang P, Luo M, 2016. Control traffic balancing in software defined networks. *Comput Netw*, 106:260-271. https://doi.org/10.1016/j.comnet.2015.08.004

Lyu N, Song GH, Yang BW, et al., 2019. QNGPSR: a Q-network enhanced geographic ad-hoc routing protocol based on GPSR. Proc 88th Vehicular Technology Conf, p.1-6. https://doi.org/10.1109/VTCFall.2018.8690651

Ma X, Xu Y, Sun GQ, et al., 2013. State-chain sequential feedback reinforcement learning for path planning of autonomous mobile robots. *J Zhejiang Univ-Sci C (Comput & Electron)*, 14(3):167-178. https://doi.org/10.1631/jzus.C1200226

Mnih V, Kavukcuoglu K, Silver D, et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533. https://doi.org/10.1038/nature14236

Peng J, Williams RJ, 1996. Incremental multi-step Q-learning. *Mach Learn*, 22(1-3):283-290. https://doi.org/10.1007/BF00114731

Shi RH, Deng YY, 2008. An improved scheme for reducing the latency of AODV in ad hoc networks. Proc 9th Int Conf for Young Computer Scientists, p.594-598. https://doi.org/10.1109/ICYCS.2008.325

Sutton RS, Barto AG, 1998. Reinforcement Learning: an Introduction. MIT Press, Cambridge, p.1.

Watkins CJCH, Dayan P, 1992. Q-learning. *Mach Learn*, 8(3-4):279-292. https://doi.org/10.1007/BF00992698

Wu C, Ohzahata S, Kato T, 2012. Routing in VANETs: a fuzzy constraint Q-learning approach. Proc Global Communications Conf, p.195-200. https://doi.org/10.1109/GLOCOM.2012.6503112

Wu C, Ohzahata S, Kato T, 2013. Flexible, portable, and practicable solution for routing in VANETs: a fuzzy constraint Q-learning approach. *IEEE Trans Veh Technol*, 62(9):4251-4263. https://doi.org/10.1109/TVT.2013.2273945

Xu DH, Chiang M, Rexford J, 2011. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Trans Netw*, 19(6):1717-1730. https://doi.org/10.1109/TNET.2011.2134866

Zhan HW, Zhou Y, 2008. Comparison and analysis AODV and OLSR routing protocols in ad hoc network. Proc 4th Int Conf on Wireless Communications, Networking and Mobile Computing, p.1-4. https://doi.org/10.1109/WiCom.2008.578

Zhang JJ, Xi K, Luo M, et al., 2014. Load balancing for multiple traffic matrices using SDN hybrid routing. Proc 15th Int Conf on High Performance Switching and Routing, p.44-49. https://doi.org/10.1109/HPSR.2014.6900880