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

Dynamic value iteration networks for the planning of rapidly changing UAV swarms*

Wei LI[1], Bowei YANG‡[1], Guanghua SONG[1], Xiaohong JIANG[2]

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

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

E-mail: li2ui2@zju.edu.cn; boweiy@zju.edu.cn; ghsong@zju.edu.cn; jiangxh@zju.edu.cn

Received Dec. 19, 2019; Revision accepted June 27, 2020; Crosschecked Oct. 20, 2020; Published online Jan. 12, 2021

Abstract: In an unmanned aerial vehicle ad-hoc network (UANET), sparse and rapidly mobile unmanned aerial vehicles (UAVs)/nodes can dynamically change the UANET topology. This may lead to UANET service performance issues. In this study, for planning rapidly changing UAV swarms, we propose a dynamic value iteration network (DVIN) model trained using the episodic Q-learning method with the connection information of UANETs to generate a state value spread function, which enables UAVs/nodes to adapt to novel physical locations. We then evaluate the performance of the DVIN model and compare it with the non-dominated sorting genetic algorithm II and the exhaustive method. Simulation results demonstrate that the proposed model significantly reduces the decision-making time for UAV/node path planning with a high average success rate.

Key words: Dynamic value iteration networks; Episodic Q-learning; Unmanned aerial vehicle (UAV) ad-hoc network; Non-dominated sorting genetic algorithm II (NSGA-II); Path planning

https://doi.org/10.1631/FITEE.1900712 **CLC number:** TP183; TP393.1

1 Introduction

Following the increasingly complex application environment of unmanned aerial vehicles (UAVs) (Bekmezci et al., 2013) and the rapid deployment of UAV ad-hoc networks (UANETs), the adaptation of artificial intelligence (AI) solutions to mobile network environments (Zhang T et al., 2017; Zhang CY et al., 2019) has become the research focus in today's information network era. An advanced UANET communication architecture is the basis for efficient communication among UAV swarms. The successful completion of a UAV mission depends on a high-quality and efficient communication network. However, sparse and rapidly mobile UAVs/nodes can dynamically change the topology of UANETs, which may critically affect UAV communication performance. Moreover, the topology of UANETs is significantly affected by every UAV mission. To minimize the impact of these factors on UANET communication and ensure better communication performance, a real-time task-network joint optimization for the UAVs is saliently required.

Conventional multi-UAV collaborative planning methods (Lee et al., 2013; Roberge et al., 2013; Koohifar et al., 2017), including genetic algorithms, specifically, the non-dominated sorting genetic algorithm II (NSGA-II) and particle swarm optimization, can solve the above-mentioned problems (Deb et al., 2002). NSGA-II tends to prematurely converge to the local Pareto-optimal front. Furthermore, NSGA-II requires that all constraints must be met to

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61501399), the SAIC MOTOR (No. 1925), and the National Key R&D Program of China (No. 2018AAA0102302)

ORCID: Wei LI, https://orcid.org/0000-0001-5102-8597; Bowei YANG, https://orcid.org/0000-0001-8581-3817

complete optimization. For example, in UANETs, there are constraints such as mobile communication interference, network connectivity, and moving obstacles. As the optimization aim increases, the training convergence speed and efficiency of NSGA-II decrease, making it impossible to achieve the real-time planning of UAVs using the algorithm.

Recently, value iteration networks (VINs) (Tamar et al., 2017), which are used to solve discrete or continuous path planning problems, have been proposed for grid map based autonomous navigation in the fields of imitation learning (Schaal, 1999) and reinforcement learning (RL) (François-Lavet et al., 2018). VINs are an approximation of the value iteration algorithm (Bellman, 1966; Bertsekas, 1995), which combines recurrent convolutional neural networks (CNNs) (Krizhevsky et al., 2017) and max-pooling (Boureau et al., 2010) to accelerate the value iteration (VI) process. VINs that are based on a CNN strategy can be effectively trained to plan several unknown static scenarios and solve complex multi-objective problems.

To improve the communication performance among dynamic UAVs, we propose a dynamic value iteration network (DVIN) model, which uses a VIN algorithm. The DVIN model is trained using episodic Q-learning (Niu et al., 2018) as per the following constraints: the real-time changing UANET topology and the anti-collision strategy of UANETs. Based on the model's training time using training samples, a state value spread function of a complete VIN in a specific flight space is obtained and represented as a two-dimensional (2D) heat map (a state value spread map) of the DVIN model. A UAV agent always moves to areas of the heat map with high intensity because these areas ensure high immediate return. In other words, the state value spread function enables the DVIN model to quickly make accurate predictions, which can help prevent UAV collisions and adjust UANET topology in real time, while the model performs its assigned tasks.

2 Related work

2.1 Reinforcement learning

RL is an abstract description of a Markov decision process (MDP) and the Bellman equation (Bellman, 1966; Bertsekas, 1995), which emphasizes how agents act based on an environment to achieve the maximum expected benefits. An MDP is a four-tuple (S, A, P, R), where S is a set of agent states in an environment, and A is a set of possible actions that an agent can take. An agent in a state $s \in S$ must select an action $a \in A$, which will produce a transition probability $P(s'|s,a)$ from state s to another state s', and an immediate reward $r = R(s,a)$ after the state transition.

VI is a typical algorithm for MDPs. We define a value $V(s)$ of a state s as the maximum reward from that state. Each VI process updates its state value $V(s)$. The $Q(s,a)$ of each action a after reaching a next state s' and receiving a reward r is calculated as follows:

$$Q_t(s,a) = r + \gamma \sum_{s'} P(s'|s,a)\, V_t(s'), \tag{1}$$

where γ is a discount factor, which reflects the importance of future rewards compared to current rewards. t is a timestamp before the agent transits to a next state s'. The best action in the action set A is selected to update the state value for an agent:

$$V_{t+1}(s) = \max_a Q_t(s,a). \tag{2}$$

During the VI process the Bellman equation is used to update the state value at each position:

$$V_*(s) = \max_a \sum_{s',r} P(s'|s,a)[r + \gamma V_*(s')], \tag{3}$$

where "$*$" represents the optimum and $V(s)$ is the current state, comprising the following two parts: the reward r after taking action a and the new state $V(s')$ reached after taking action a.

Q-learning (Watkins and Dayan, 1992) is a classic RL algorithm. The essence of Q-learning is that the current state s, action a, reward r, and the Q function have the following relationship:

$$Q_t(s,a) = r_t + \gamma \max_a Q_{t+1}(s,a). \tag{4}$$

In Q-learning, a Q-table is updated using Eq. (4) and used to map states and actions to Q-values. When using a greedy policy, an agent will select an optimal action for the current state s using the equation below:

$$a^* = \arg\max_{a \in A} Q_t(s,a). \tag{5}$$

Mnih et al. (2015) proposed a deep Q-network (DQN) to enable agents to extract important features in an environment from high-dimensional inputs and use these features to generalize experience to new situations. The DQN uses multi-layer perceptions as a function approximator to estimate the action-value function.

Zhang CY et al. (2019) thoroughly reviewed the embedding of deep learning and RL into mobile and wireless networks. Based on RL, several algorithms have been derived to solve problems such as trajectory optimization of multiple cellular-connected UAVs (Challita et al., 2018).

2.2 Value iteration network for 2D maze domain navigation

A VIN is a general policy representation that learns and solves unknown MDPs. Embedding a planning model into a deep neural network (NN) and training the NN using an end-to-end training method will make the trained network (a VIN) to be able to plan in a decision-making process. The trained VIN can be used to generalize different and invisible scenarios such as 2D maze domain autonomous navigation.

In a 2D maze domain with randomly placed obstacles, the position of an agent can be identified, as well as the map of the obstacles and the goal position. By learning the optimal strategies in a considerable number of 2D maze domains or using an RL training method, VIN models can learn to solve novel tasks.

3 Problem description and dynamic value iteration network model

3.1 UANET problem description

To reduce the dependence of collaborative communication among UAVs on basic communication facilities, such as ground stations or satellites, UANETs use UAVs as nodes in their networks. Each node (i.e., a UAV) sends and receives data from each other, automatically connecting to build mobile multi-hop wireless networks. If UAVs/nodes can adapt to novel physical locations, the quality of the communication services among UAVs would be considerably improved. Therefore, it is essential to take complete advantage of DVINs for network communication path-finding.

In a UANET, we map sparse, rapidly mobile UAV nodes and interference obstacles into a 2D maze to represent the observation map at any given moment. The observation map includes UAVs' positions (denoted by blue nodes) and obstacle positions (denoted by black nodes) in the UANET (Fig. 1). The blue nodes belong to ad-hoc networks. The blue nodes must ensure their network connectivity and avoid obstacles in a UANET to complete their respective tasks. A green node is used to indicate the start position of a UAV/node whose mission is to reach the goal position denoted by a red node. Changes in the topology of UANETs are based on the communication range and mobility of each UAV/node. Therefore, for UAVs to complete their missions, they must identify an optimal path, which is shown as the black path in Fig. 1.

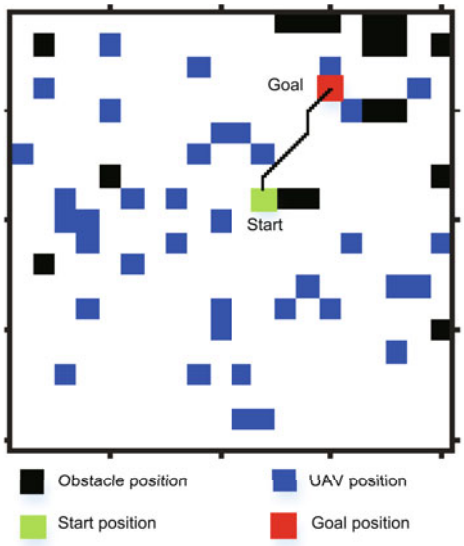

Fig. 1 Sample data of 2D mazes: an optimal path of one UAV moving from the start position to the goal position (References to color refer to the online version of this figure)

Furthermore, a connectivity goal map is generated from the observation map, which is a one-hot representation of the aim ("1" represents a non-test UAV/node that can communicate with the test UAV/node, and "0" otherwise).

For the UANET, let $N = \{1, 2, ..., n\}$ represent the set of UAVs/nodes in the network where n means the total number of UAVs/nodes in the UANET. con_count is the number of nodes in set N that can directly or indirectly communicate with other nodes

but itself. The network connectivity (NC) of each node is then defined as follows:

$$\mathrm{NC} = \frac{\mathrm{con\_count}}{n}. \tag{6}$$

Therefore, the full network connectivity (FNC) of the UANET is as follows:

$$\mathrm{FNC} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{NC}_i, \tag{7}$$

where FNC is the average connectivity performance obtained by agents at their current positions, whose value is between 0 and 1.

The movement of each node in the UANET rapidly changes the FNC value, and the FNC value reflects the average network connectivity performance. The FNC value at the next moment is related only to the state of the UAVs/nodes at the current moment. Therefore, the change of the FNC value completely conforms to an MDP and the Bellman equation. Hence, we propose the DVIN model.

3.2 DVIN model

The purpose of using DVINs is to realize real-time re-planning of a single UAV agent's path in rapidly changing UAV scenarios, thus ensuring reliable communication between any two nodes and maximizing the FNC value in the UANET. Compared to a previous study (Tamar et al., 2017) using VINs, DVINs can be used to train models in dynamic scenarios. The architecture of DVINs is shown in Fig. 2.

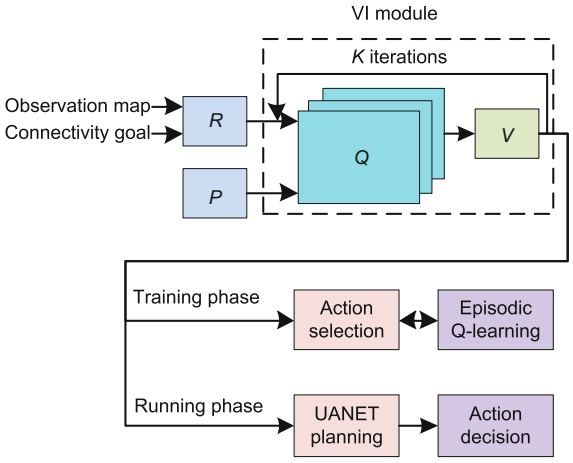

Fig. 2 Architecture of a dynamic value iteration network (DVIN)

First, we concatenate the observation map data with a connectivity goal map. Second, we map the concatenated data into a reward map R through two convolution operations (Krizhevsky et al., 2017). Third, we input the reward map R and the state transition probability P into the VI module (Tamar et al., 2017) and obtain the state value map V. In the action selection (AS) module, the state value map V is considered as an instructive state value function, which is based on the ϵ-greedy policy (Tokic and Palm, 2011) to select an action. Finally, in continuously moving scenarios, any selected action is trained using a trial-and-error method and backpropagation by episodic Q-learning to obtain the training parameters θ of the DVIN model. In the testing phase, parameters θ are used for node path planning and final action decision-making for the UANET.

3.2.1 Value iteration module

The VI module is an approximation of the VI algorithm for K iterations (Tamar et al., 2017), which can be represented by a specific CNN architecture. The VI algorithm follows an MDP and the Bellman equation, which is defined in iterative processes (Q function and V function) as Eqs. (1) and (2).

An agent in a state s must select an action a, which will provide a reward $R(s,a)$ and give the next state s' with the transition probability $P(s'|s,a)$. In the VI module, the inputs R and P are equivalent to $R(s,a)$ and $P(s'|s,a)$, respectively. We then stack the reward map R with the state value map V from the previous iteration to generate a Q map. The state value map V (corresponding to $V_{t+1}(s)$) provides a maximum pooling operation, which is produced from the Q map. After K iterations, the final state value map V is obtained (Fig. 3).

3.2.2 Action selection module

In the AS module, the state value map V can be considered as a 2D heat map in DVINs where each grid represents a state value. The higher the state value of a grid, the closer an agent tends to move to that grid. This indicates that agents in the heat map always move to grids with high state values because grids with high state values yield higher immediate returns.

For each trained 2D maze, using one of the UAVs/nodes in Fig. 1 as an example, (x,y) is the

current position of this node. The node positions in different instances will change the FNC value. To maximize the FNC value in the UANET, a movement technique is used by the DVIN model to select the next position for a certain node, as described in Algorithm 1.

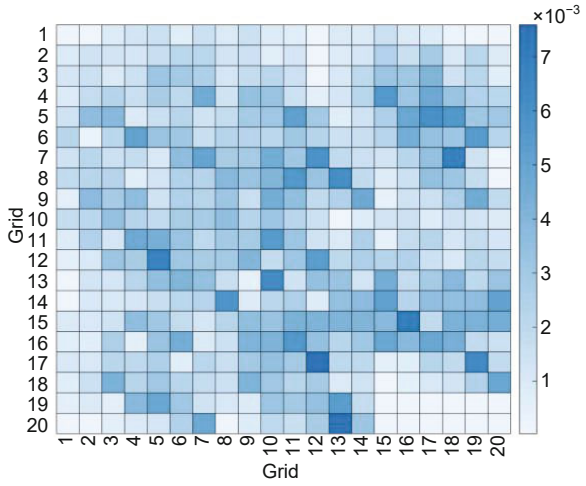

Fig. 3 Visualization of the corresponding final state value map of Fig. 1 after training. Each grid is represented by a value V

In the training phase, the FNC value should exceed the network connectivity requirement threshold we pre-set. Moreover, V is converted to a pseudo action value map, representing the action value moving from any start position in the 2D maze to one of the elements in set poss. Each position corresponds to nine action values, which form the action set A. This indicates that the UAV agent will move to a position where the FNC value can be maximized as much as possible. A final action probability list p_{poss} for the set A is obtained in a softmax policy (Cruz et al., 2019) as Eq. (8), and the movement of the agent from the current position to the next position depends on the maximum probability of list p_{poss}.

$$p_{\text{poss}}(a)=\frac{\exp(V_{\text{poss}}(a)/\tau)}{\sum\limits_{a_i\in A}\exp(V_{\text{poss}}(a_i)/\tau)}, \tag{8}$$

where τ is a positive parameter called temperature.

To completely reflect the agents' autonomy in exploring unknown environments, an ϵ-greedy policy is introduced when selecting actions. The ϵ-greedy policy as Eq. (9) is used to balance exploitation and exploration:

$$a^*=\begin{cases}\arg\max_{a\in A} Q_t(s,a), & \text{with probability } 1-\epsilon,\\ \text{random}(A), & \text{with probability } \epsilon.\end{cases} \tag{9}$$

For the ϵ-greedy strategy, the probability that the agent has an ϵ value will be used for the exploration operation; however, the probability $1-\epsilon$ value will be used for the exploitation operation, for which an action a is optimally selected from the action set A as the next step. The exploitation operation can then be defined as Eq. (5).

Algorithm 1 Next-action selection process of DVINs

Input: current position in 2D maze (x,y), set of neighbors' positions (including its own position) poss, and the state value map V
Output: next action a^*

$(x,y)\rightarrow$ FNC
if FNC $<$ threshold **then**
  if Phase is training **then**
    Select a^* with the ϵ-greedy policy based on poss and V
    Update (x,y) to (x',y') based on a^*
    $(x',y')\rightarrow$ FNC
  else
    $p_{\text{poss}}=\text{softmax}(V)$
    Select a^* greedily according to the probability p_{poss}
  end if
else
  Stay in current position (x,y) or perform its own task
end if

3.2.3 Episodic Q-learning

Episodic Q-learning is unlike the original n-step Q-learning (Mnih et al., 2016; Niu et al., 2018), where each episodic terminates when an agent reaches an aim or the maximum step threshold is reached. In episodic Q-learning, a UAV agent will continuously move from s_t to s_{t+1} following the action selected from the AS module until the state reaches any of the following conditions: the FNC value reaches the threshold value which we pre-set; the UAV agent moves to obstacle positions; the UAV agent oversteps the boundary of the 2D maze; the UAV agent achieves the longest step we pre-set in the 2D maze. Subsequently, the trainable weights after the entire

episode are updated. For each episode, the loss function is then defined as follows:

$$L(\theta)=\sum_{t=1}^{T}\left(R_t-Q_{s_t}\right)^2, \tag{10}$$

where

$$Q_{s_t}=Q(s_t,a_t;\theta_t), \tag{11}$$

R_t is the expected return, θ_t is the weight of the DVIN, Q_{s_t} is a state value function in the DVIN, and T means the total number of steps of each episode. To use the loss function (Eq. (10)), R_t is defined as follows:

$$R_t=R+\gamma R_{t+1}, \tag{12}$$

where the reward R is an immediate return after performing one step in an episode. When the training procedure stops, the optimal action (Eq. (5)) is determined.

3.2.4 Running phase

To plan for the rapidly changing UAV scenarios, the running phase is divided into the UANET planning (UP) module and action decision (AD) module.

In the UP module, we use the DVIN's training parameters θ from the training phase to predict a position in real time where the agent should go at the next moment. An optimal action is then obtained by greedily selecting actions from the state value map V rather than the ϵ-greedy policy, as shown in Algorithm 1. In the AD module, the optimal and instructive action is obtained from the UP module, which every UAV in the UANET should use to re-plan its path in real time, thus ensuring the best connectivity performance in the network.

4 Performance simulation

It is difficult to directly model flight trajectories in the UANET because a UAV's flight trajectory and real flight scenes are uncertain and complicated. To adequately simulate UAV mobility patterns, the flight trajectory data around Paris' Charles de Gaulle Airport are collected from flightradar24.com. Moreover, four typical mobility patterns of UAV are shown as examples in Fig. 4.

As per the mobility patterns in commercial flight data, we process these data and use the processed movement trajectory data to train the proposed DVIN model. Therefore, the collected region size, moving speed, and the number of nodes are rescaled to satisfy the UAVs' movement constraints in our UANET scenarios. The positions of these movement trajectories at each moment are then mapped into a single 2D maze (Fig. 1), where UAV flight obstacles outside the trajectory are artificially set in proper positions. Because the preprocessed data can represent real-world UAVs' mobility, we use Mininet-WiFi (Fontes et al., 2015) to simulate the network environment of a UANET. The Mininet-WiFi emulator is an extension of the existing Mininet (Fontes et al., 2015), which is a platform for network development and testing of software-defined networks.

One of the 2D mazes is emulated (Fig. 5), where the FNC value can be calculated. In this scenario, a UANET is formed by 50 UAVs/nodes defined as sta0–sta49 for simplicity while programming. The wmediumd method (Fontes, 2019) is then used to simulate wireless media, the logDistance model (Fontes, 2019) is used as a propagation model, and the communication range between nodes is set to six grids. By setting these parameters, we generate 108 000 2D mazes for training and testing our DVIN model. These algorithms, the NSGA-II and exhaustive method (EM), are then used as the benchmark methods to evaluate the DVIN model's performance.

The EM is used to calculate the highest FNC value that an agent can achieve per episode. The FNC value calculated based on the EM demonstrates the optimal network performance of the UANET. Moreover, we calculate the time cost of each episode under EM to measure the performance of the DVIN and NSGA-II algorithms.

First, before the running phase, the DVIN model is trained to obtain the training parameters θ. In episodic Q-learning, we set the discount factor $\gamma=0.99$, and set the reward $R=1$ if FNC is greater than or equal to the threshold. We then set $R=-1$ if the agent's next step is in an obstacle position or oversteps the boundary of the 2D maze. Otherwise, we set $R=-0.05$, indicating that the agent is given a small penalty value and has a greater tendency to move to a position with greater returns in the next selection operation. We then set $\epsilon=0.2$ for the ϵ-greedy policy. We use the Adam optimizer, whose

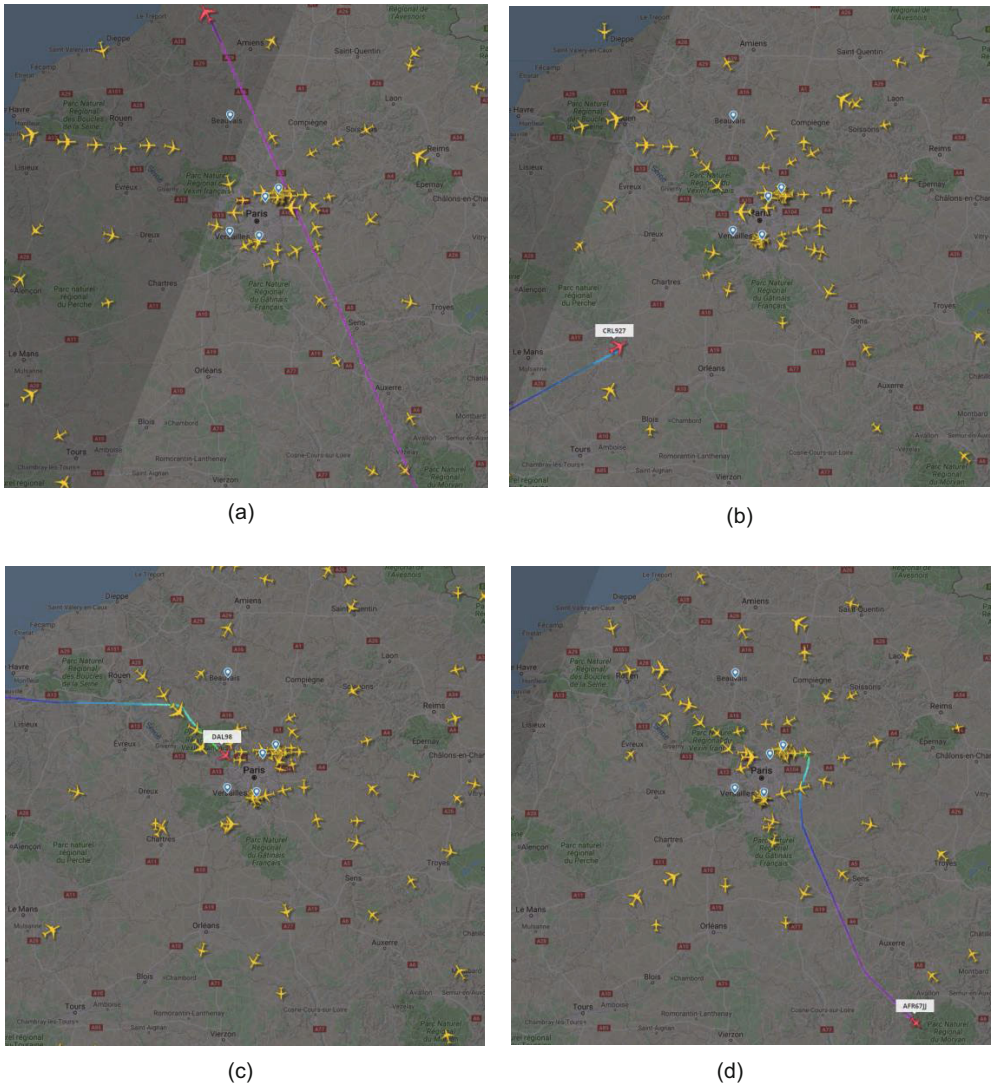

Fig. 4 Four typical mobility patterns of UAVs: (a) a node goes straight through the entire map area, and its direction is roughly from south to north; (b) a node flies into the map area but is not ready to land; (c) a node flies into the map area and prepares to land; (d) a node takes off from a position in the map area and prepares to fly out of the map area

initial learning rate is 0.01 as the training optimizer for the model. We then set the number of iterations K to 40, which represents the number of times the value is iterated in the VI module.

Furthermore, to compare the performance of the DVIN algorithm, NSGA-II, and EM, we set a primary training parameter: the threshold is set to 100%, which indicates that the agent node is connected to the whole network in the UANET and that the network communication performance is the optimal.

Compared with the conventional NSGA-II and EM, the AI algorithm, which is the DVIN algorithm, uses graphics processing units (Buck et al., 2004) to train the DVIN model and speed up the model's convergence. Hence, we train the DVIN model on a single machine with the NVIDIA GeForce GTX 1080Ti graphics card, and this is facilitated using the Tensorflow framework (Abadi et al., 2016). After training the model, the average loss of 32 epochs is obtained (Fig. 6), and the average reward value for the agent varies based on the training epochs (Fig. 7). It can be seen that the reward value increases as the number of training epochs increases until the training approaches convergence. We then train each epoch with 4440 episodes with an average training

time of 5364.14 s at 17 steps per episode. For each episode, we detect the state of the agent (a certain UAV test node) at a current position in real time. If FNC<threshold, we require to re-plan the position of the agent to achieve an FNC value that is equal to the threshold value we pre-set. After re-planning, if FNC≥threshold, the episode's path planning is considered a success; otherwise, it is considered to be unsuccessful, and the training success rate is 38.06%.

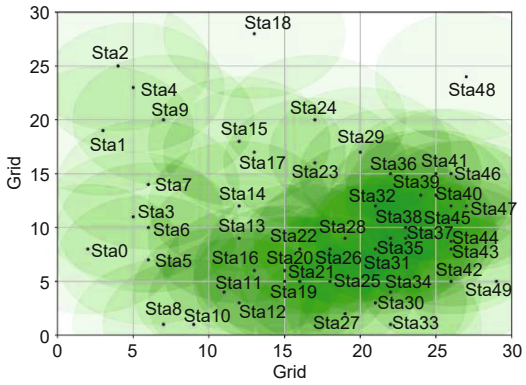

Fig. 5 A simulation result of a 2D maze for UAV swarms

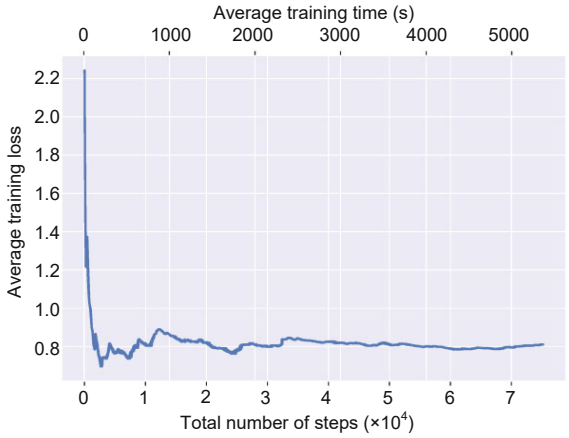

Fig. 6 Average loss of the training epochs (75 160 steps, 5364.14 s per epoch)

Fig. 7 Average reward value of each training epoch for DVIN

In the testing phase, to avoid overfitting and pseudo-high success rates, testing episodes that are unlike episodes used in the training phase are selected as test scenarios. We test a total of 1615 episodes, thus achieving a success rate of 34.30%. Moreover, the agent makes an average of 16 steps and 1.19 s per episode, and these test scenarios are used as experimental data for NSGA-II and EM.

We then execute all of the algorithms on the same hardware device (Table 1). The device configuration is a single machine with Intel Core i7-7700K 4.2 GHz processor, 4 cores, as well as 16 GB memory. The success rate of EM is 42.81%, which is the highest success rate using the experimental data, but the average time per episode is 81.31 s. NSGA-II has a success rate of 30.41% and an average time per episode of 43.98 s.

Table 1 Performance comparison using different methods

Algorithm	Success rate (%)	Average time per episode (s)
DVIN	34.30	1.19
NSGA-II	30.41	43.98
EM	42.81	81.31

Because of the rapidly changing UANET topology, our experimental results demonstrate that the NSGA-II algorithm cannot realize the requirement for agents to achieve real-time path re-planning when the FNC value is set too high. The proposed DVIN model with a higher accuracy considerably reduces the decision-making time for path planning. Compared with NSGA-II, our model achieves a speed 37 times greater than that of NSGA-II.

Furthermore, as shown in Fig. 8, we demonstrate the impact of different parameters, including the UAV's communication range and the network connectivity requirement threshold, on the DVIN model. The following conclusions can be drawn:

1. To achieve real-time network connectivity, the communication range of each node in the UANET plays an important role.

2. The smaller the threshold value, the better the performance of the DVIN model.

3. In real UANET scenarios, it is inappropriate to set the threshold to a minimum value when the

communication range of the UAVs is large.

The experimental results demonstrate the effectiveness of the DVIN model and guide the deployment of the DVIN model in real UANET scenarios. In a UANET, it is difficult to achieve full connectivity. However, using the DVIN algorithm, we can identify an optimal trajectory in 1 or 2 s. Compared to conventional optimization or planning algorithms, the DVIN algorithm is more appropriate to use in rapidly changing networks.

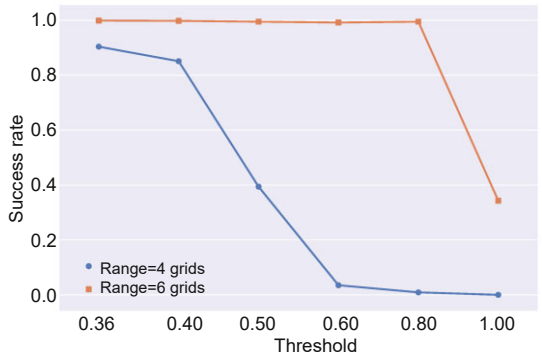

Fig. 8 Effect of different threshold settings on the success rate of DVIN

5 Conclusions

In this study, we have proposed a DVIN model to plan the movement of UAVs in rapidly changing environments. Simulation results demonstrated that the DVIN model achieves higher real-time re-planning efficiency compared to models using NSGA-II. Although the DVIN model has a greater training time, the decision-making time in the running phase is considerably less than that of NSGA-II. Moreover, in the running phase, the average success rate of each episode of the DVIN model is higher than that of models using NSGA-II. In this study we have examined only the path planning problem of a single UAV agent. The real-time task planning problem of multi-agent collaborative control will be examined in our future research.

Contributors

Wei LI and Bowei YANG designed the research. Wei LI processed the data and drafted the manuscript. Guanghua SONG and Xiaohong JIANG helped organize the manuscript. Wei LI and Bowei YANG revised and finalized the paper.

Compliance with ethics guidelines

Wei LI, Bowei YANG, Guanghua SONG, and Xiaohong 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.

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

Bellman R, 1966. Dynamic programming. *Science*, 153(3731):34-37. https://doi.org/10.1126/science.153.3731.34

Bertsekas DP, 1995. Dynamic Programming and Optimal Control. Athena Scientific, Belmont, USA.

Boureau YL, Bach F, LeCun Y, et al., 2010. Learning mid-level features for recognition. Proc IEEE Computer Society Conf on Computer Vision and Pattern Recognition, p.2559-2566. https://doi.org/10.1109/CVPR.2010.5539963

Buck I, Foley T, Horn D, et al., 2004. Brook for GPUs: stream computing on graphics hardware. *ACM Trans Graph*, 23(3):777-786. https://doi.org/10.1145/1015706.1015800

Challita U, Saad W, Bettstetter C, 2018. Deep reinforcement learning for interference-aware path planning of cellular-connected UAVs. Proc IEEE Int Conf on Communications, p.1-7. https://doi.org/10.1109/ICC.2018.8422706

Cruz F, Wüppen P, Fazrie A, et al., 2019. Action selection methods in a robotic reinforcement learning scenario. Proc IEEE Latin American Conf on Computational Intelligence, p.1-6. https://doi.org/10.1109/LA-CCI.2018.8625243

Deb K, Pratap A, Agarwal S, et al., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput*, 6(2):182-197. https://doi.org/10.1109/4235.996017

Fontes RR, 2019. Emulando Redes Sem Fio Com Mininet-WiFi. https://github.com/ramonfontes/mn-wifi-book-pt/blob/master/preview-book.pdf

Fontes RR, Afzal S, Brito SHB, et al., 2015. Mininet-WiFi: emulating software-defined wireless networks. Proc 11th Int Conf on Network and Service Management, p.384-389. https://doi.org/10.1109/CNSM.2015.7367387

François-Lavet V, Henderson P, Islam R, et al., 2018. An introduction to deep reinforcement learning. *Found Trends® Mach Learn*, 11(3-4):219-354. https://doi.org/10.1561/2200000071

Koohifar F, Kumbhar A, Guvenc I, 2017. Receding horizon multi-UAV cooperative tracking of moving RF source. *IEEE Commun Lett*, 21(6):1433-1436. https://doi.org/10.1109/LCOMM.2016.2603977

Krizhevsky A, Sutskever I, Hinton GE, 2017. ImageNet classification with deep convolutional neural networks. *Commun ACM*, 60(6):84-90. https://doi.org/10.1145/3065386

Lee J, Kang BY, Kim DW, 2013. Fast genetic algorithm for robot path planning. *Electron Lett*, 49(23):1449-1451. https://doi.org/10.1049/el.2013.3143

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

Mnih V, Badia AP, Mirza L, et al., 2016. Asynchronous methods for deep reinforcement learning. Proc 33rd Int Conf on Machine Learning, p.1928-1937.

Niu SF, Chen SH, Guo HY, et al., 2018. Generalized value iteration networks: life beyond lattices. Proc 32nd AAAI Conf on Artificial Intelligence, p.6246-6253.

Roberge V, Tarbouchi M, Labonte G, 2013. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Trans Ind Inform*, 9(1):132-141. https://doi.org/10.1109/TII.2012.2198665

Schaal S, 1999. Is imitation learning the route to humanoid robots? *Trends Cogn Sci*, 3(6):233-242. https://doi.org/10.1016/s1364-6613(99)01327-3

Tamar A, Wu Y, Thomas G, et al., 2017. Value iteration networks. Proc 26th Int Joint Conf on Artificial Intelligence, p.4949-4953. https://doi.org/10.24963/ijcai.2017/700

Tokic M, Palm G, 2011. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. Proc 34th Annual German Conf on Advances in Artificial Intelligence, p.335-346. https://doi.org/10.1007/978-3-642-24455-1_33

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

Zhang CY, Patras P, Haddadi H, 2019. Deep learning in mobile and wireless networking: a survey. *IEEE Commun Surv Tutor*, 21(3):2224-2287. https://doi.org/10.1109/COMST.2019.2904897

Zhang T, Li Q, Zhang CS, et al., 2017. Current trends in the development of intelligent unmanned autonomous systems. *Front Inform Technol Electron Eng*, 18(1):68-85. https://doi.org/10.1631/FITEE.1601650