1-2023

Intelligent adaptive gossip-based broadcast protocol for UAV-MEC using multi-agent deep reinforcement learning

Zen REN

Xinghua LI

Yinbin MIAO

Zhuowen LI

Zihao WANG

See next page for additional authors

Citation

REN, Zen; LI, Xinghua; MIAO, Yinbin; LI, Zhuowen; WANG, Zihao; ZHU, Mengyao; LIU, Ximeng; and DENG, Robert H.. Intelligent adaptive gossip-based broadcast protocol for UAV-MEC using multi-agent deep reinforcement learning. (2023). *IEEE Transactions on Mobile Computing*. 1-17.
Available at: https://ink.library.smu.edu.sg/sis_research/8275

DOI: 10.1109/TMC.2023.3323296

Intelligent Adaptive Gossip-based Broadcast Protocol for UAV-MEC using Multi-agent Deep Reinforcement Learning

Zhe Ren, Xinghua Li, *Member, IEEE*, Yinbin Miao, Zhuowen Li, Zihao Wang, Mengyao Zhu, Ximeng Liu, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

Abstract—UAV-assisted mobile edge computing (UAV-MEC) has been proposed to offer computing resources for smart devices and user equipment. UAV cluster aided MEC rather than one UAV-aided MEC as edge pool is the newest edge computing architecture. Unfortunately, the data packet exchange during edge computing within the UAV cluster hasn't received enough attention. UAVs need to collaborate for the wide implementation of MEC, relying on the gossip-based broadcast protocol. However, gossip has the problem of long propagation delay, where the forwarding probability and neighbors are two factors that are difficult to balance. The existing works improve gossip from only one factor, which cannot select suitable forwarding probability and avoid redundant messages. Besides, these schemes do not consider the historical packet reception of new neighbors when UAVs fly around, which decreases forwarding efficiency. To solve these problems, we first propose a data structure called Bitgraph that can record the historical packet reception of UAVs. Then, we formulate gossip broadcasting as a partially observable Markov decision process. Based on Bitgraph, we design the reward function. Finally, we design a multi-agent reinforcement learning algorithm, Branching Deep Graph Network (BDGN), which simultaneously makes decisions on forwarding probability and neighbors. Extensive experiments illustrate that our proposal gets more than 29% advantage in terms of the propagation delay and 20% advantage in terms of the redundant messages compared to the existing works.

Index Terms—UAVs, Gossip Protocol, Sparse Rewards, Partially Observable Markov Decision Process, Reinforcement Learning.

1 Introduction

With the emergence of smart navigation, online 3A gaming and virtual reality, the convenience of mobile edge computing (MEC) has attracted lots of attention. The global edge computing market is expected to grow at a compound annual growth rate of 37.9% from 2023 to 2030 to reach USD 155.90 billion by 2030 [1]. Recently, UAV-assisted mobile edge computing (UAV-MEC) has become a new style of MEC to provide more flexible, easier and faster computing service than traditional fixed location MEC infrastructures. Past works [2], [3], [4], [5] have considered a single UAV case. Unfortunately, one UAV can only provide service for limited users. Recent work [6], [7], [8] has evolved to where UAV clusters act as edge pools to provide edge computing services. For example, multiple UAVs hovering in the air provide computational services for underwater and surface sensors in a marine network [6]. However, the above schemes only focus on optimizing the energy consumption of the sensors and UAVs during task offloading. They do not consider the data packet exchange process during edge computing within the UAV cluster. The data packet exchange of UAV clusters can refer to distributed computing systems, relying on some broadcast protocol [9].

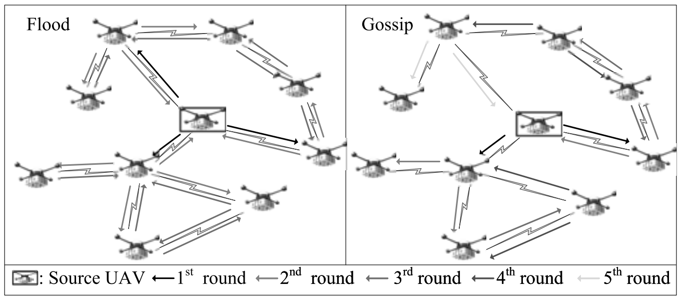


Fig. 1. One UAV is used as the source, and the broadcasting scenario under flood and gossip is adopted. The forwarding strategy of gossip is configured with a forwarding probability of 80%, and the number of forwarding neighbors is 2. The flood can cover all UAVs with only three rounds of propagation, while gossip needs five rounds to cover all UAVs. At the same time, the number of message packets is represented by arrows. Compared with the flood, gossip reduces the number of redundant messages by 53.8%.

• *Zhe Ren, Xinghua Li, Yinbin Miao, Zhuowen Li, Zihao Wang and Mengyao Zhu are with State Key Laboratory of Integrated Services Networks, and the School of Cyber Engineering, Xidian University, Xi'an, 710071, China. Xinghua Li is also with Engineering Research Center of Big data Security, Ministry of Education, Xi'an 710071, China (E-mail: jerryren2884@gmail.com; xhli1@mail.xidian.edu.cn; ybmiao@xidian.edu.cn; zw_li@xidian.edu.cn; wangzihao318@mail.xidian.edu.cn;zhumengyao@xidian.edu.cn;).*
Ximeng Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350108, China (e-mail: snbnix@gmail.com).
Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 178902 (e-mail: robert-deng@smu.edu.sg).

(Corresponding Author: Xinghua Li)

Existing broadcast protocols include flood and gossip [10]. The advantage of the flood is that the propagation delay is optimal, but the problem is message implosion [11]. In contrast, gossip can avoid implosion, which is more suitable for UAVs with limited energy [12]. Each UAV, as a source, initially sends its message packet to several neighbors. In each subsequent round, the relay UAV must decide whether to forward the packet with a certain probability when receiving the message packet. If it forwards, it randomly selects neighbors to forward. Theoretically, gossip will run for multiple rounds until all UAVs receive the message packet. However, as shown in Fig. 1, due to the random forwarding characteristics, there is a problem of long propagation delay [13]. To reduce the propagation delay, the existing works improve the gossip from forwarding probability decision making [14], [15] and forwarding neighbors decision making [16], [17], [18], [19], [20].

The forwarding probability and neighbors of the gossip protocol are two factors that are difficult to balance. When the forwarding probability is selected as 100%, and the forwarding neighbors are all neighbors, the gossip degenerates into a flood. However, existing works improving gossip only focus on one single aspect. The schemes [14], [15] that only consider the forwarding probability is still completely random when selecting neighbors, which will cause the message packet to be sent to the node that has already received it. This situation will cause this round of forwarding to be invalid, so it cannot guarantee a lower propagation delay. The schemes [16], [17], [18], [19], [20] that only consider the forwarding neighbors introduce some fixed strategies into the decision making of forwarding neighbors. The selection of neighbors is not completely random but is based on factors such as distance and topology to select the neighbor nodes that need to receive packets. Still, these schemes do not consider the forwarding probability. Under this condition, the relay UAV still performs the same processing when receiving duplicate messages, resulting in more redundant message packets. In addition, these schemes [14], [15], [16], [17], [18], [19], [20] do not consider the neighbors' historical packet reception because getting the historical record is hard in dynamic distributed MEC environment. The dynamic change of the topology causes the neighbors around the UAV to change. Finally, these schemes do not distinguish between old and new neighbor nodes, which will cause the relay node to send redundant message packets to nodes that have already received the packet. This situation will cause a decrease in propagation efficiency, thereby prolonging the propagation delay.

Since the UAV needs to decide the forwarding probability and neighbors in multiple rounds, the gossip improvement is a sequential decision problem [21]. For such problems, reinforcement learning is a recognized solution, but there are two challenges to its use in gossip broadcasting.

- *First, in each round of gossip broadcasting, UAVs do not know the impact of the current decision for the final propagation delay. Due to the reward of final propagation delay obtained when gossip broadcasting stops, the UAVs cannot learn the forwarding strategy without extra help, which is a sparse reward problem [22]. We can design some rewards to help the UAVs learn the forwarding strategy. However, the design relies on domain knowledge, which is a hard problem.*
- *Second, the two actions of forwarding probability and neighbors need to be decided simultaneously, so there is a problem of combinatorial action space explosion [23]. At the same time, the dynamic changes of neighbors make UAV hard to learn the latent influence for action decisions between UAVs.*

To address the above challenges, we first construct a data structure Bitgraph that can record the packet reception of UAVs in a dynamic distributed environment. With the help of Bitgraph, we design a new reward function through reward reshaping, helping the UAV to learn the better forwarding policy in multiple rounds of propagation. Then, we formulate gossip broadcasting as a partially observable Markov decision process. Finally, we propose a multi-agent deep reinforcement learning (MDRL) algorithm named BDGN by innovatively combining the graph attention mechanism and branching structure network. Compared to conventional MDRL, BDGN can make decisions on forwarding probability and neighbors simultaneously, and consider the two-hop dependencies between the UAV and its neighbors in a dynamic topology environment. Our specific contributions are as follows:

1) We propose Bitgraph to record packet reception by UAVs and design a reward function based on it. Bitgraph is a customized binary string and is updated by gossip about gossip. Through reward reshaping [24], we develop Bitgraph filling percentage and message redundancy in a single round as the intrinsic reward of the UAV to guide each round of gossip forwarding decision, and we also consider the final propagation cost to help UAV learn the optimal strategy.
2) We design a multi-agent deep reinforcement learning algorithm Branching Deep Graph Network (BDGN). Specifically, we use different actor networks to decide on different actions and reduce the dimensionality of the combined action space from $O(n^2)$ to $O(n)$. Meanwhile, to cope with a dynamic topology environment, we use a deep graph neural network to learn the dependencies between the UAV and its new neighbors.
3) Using Gym's code framework, we implement a prototype system for UAV broadcast with the network simulator NS3 and the machine learning library Pytorch. Experimental results show that our scheme reduces the propagation delay by 29.2% and the number of redundant messages by 20.7% compared to the existing works.

The rest of this paper is organized as follows. First, we review related work of gossip improvement in Section 2. Then, we introduce preliminaries in Section 3. Section 4 presents the problem formulation. Section 5 describes our proposal for the intelligent adaptive gossip-based broadcast protocol. In Section 6, we use NS3 and Pytorch to conduct experiments on the proposed scheme and analyze the experimental results. Finally, the conclusion is given in Section 7.

2 Related Work

Numerous studies [10], [12] revealed gossip-based broadcast performance is related to the topology, and adjusting the selection strategy of forwarding probability and neighbors can improve the performance of gossip protocols.

2.1 Improvement on forwarding probability selection

Kyasanur *et al.* [14] proposed an adaptive gossip approach named smart gossip. This work aims to choose the correct value for every node based on network topology in a wireless sensor network. Smart gossip quantifies the importance of a node using a distributed algorithm that takes into account the local topological properties around a node. Specifically, a node chooses its gossip probability according to how many other nodes depend on this node for the reception of a gossip message. The biggest drawback of this scheme is that dynamic changes in topology can destroy the relationship between nodes, so this scheme is only suitable for static networks.

To improve the efficiency of the gossip, Cheng *et al.* [15] proposed a Binary Exponential Backoff Gossip (BEBG) algorithm that combines the binary exponential backoff algorithm with the random gossip algorithm. The message propagation strategy of BEBG is that the more times a node receives the same message, the lower the probability of continuing to propagate that message. Besides, the combination of the PULL mechanism makes BEBG achieve redundant message drop while reducing the propagation delay. Although this scheme does not require much complex computation, it is not given at what decreasing gradient the probability should be in different cases.

However, these schemes [14], [15] of forwarding probability selection are still completely random in selecting neighbors and do not guarantee an ideal propagation delay.

2.2 Improvement on forwarding neighbors selection

Tian *et al.* [16] aimed to improve the reliability and efficiency of gossip by using the bionic attractor selection model with a strong premise that the location and speed of vehicles are used as auxiliary information to make forwarding neighbor decisions. Matos *et al.* [17] proposed an enhanced version of the gossip protocol called Brisa, which discovers special structures such as trees, directed acyclic graphs, and forests under the current topology by maintaining a view called HyParView, and achieves more efficient propagation based on these structures. However, it will cost much time to find these special structures. Ozkasap *et al.* [18] proposed a hierarchical version of the gossip protocol called ProFI. ProFI divides the hierarchy containing peer nodes by constructing dominating sets and using an energy consumption model to make some nodes switch to the idle state, thus eliminating some unnecessary gossip forwarding operations.

Esposito *et al.* [19] achieved the improvement effect by introducing some determinism in gossip forwarding neighbor selection by introducing a policy learning mechanism on the basis of the proposed utility function that allows each node to efficiently determine the best forwarding neighbor node based on the actions taken by other nodes and the loss patterns in the system. However, this solution lacks some of the advantages of a fixed strategy, such as the inability to guarantee a 100% propagation coverage. Altoaimy *et al.* [20] proposed two improved versions of gossip protocols based on distance measurement. The first proposed protocol NNGossip uses the nearest neighbor distance measurement while the second proposed protocol CBGossip uses city block distance measurement to calculate the scores of the neighbor nodes to be selected by different methods, and based on the highest scores, the forwarding neighbor selection is made. Despite the simplicity of the scheme, it still requires some additional information about location and power to be entered when calculating forwarding neighbors.

However, these schemes [16], [17], [18], [19], [20] of forwarding neighbors selection still perform equal processing when receiving duplicate messages, resulting in more redundant message packets in the system.

2.3 Customized improvements made for blockchain

In addition to the two aforementioned approaches, there is a category of work that is customized for blockchain improvements. Berendea *et al.* [25] made four enhancements to gossip, which are infect upon contagion, digests for the push phase, randomization of the initial gossiper and removal of the pull component. The main idea of work [25] is to simplify and merge some operations, thus improving the efficiency of gossip. Cason *et al.* [26] presented Semantic Gossip, a method that enhances traditional gossip with two techniques: semantic filtering and semantic aggregation. Semantic filtering lets the gossip layer discard unnecessary messages based on consensus logic. On the other hand, semantic aggregation allows processes to combine multiple messages into a single message that carries the same meaning as the consensus. Adopting the two semantic techniques improves the latency of gossip-based Paxos from 7% to 24%. Xu *et al.* [27] introduced the idea of density clustering to propose the DC-Gossip broadcast protocol, constructing a stable network architecture with highly dense connectivity for the blockchain network layer. This architecture can effectively reduce the propagation latency and ensure the integrity of the distributed ledger. Saldamli *et al.* [28] proposed a predictable correction algorithm and a checking algorithm that allows gossip to reduce the number of redundant messages while maintaining propagation efficiency.

TABLE 1
A COMPARATIVE SUMMARY OF GOSSIP IMPROVEMENT SCHEMES

Schemes	Design Factors	Method	Function	
			F1	F2
Kyasanur *et al.* [14]	Forwarding probability	Distributed algorithm	✗	✓
Cheng *et al.* [15]	Forwarding probability	Fixed strategy	✓	✓
Tian *et al.* [16]	Forwarding neighbors	Heuristic algorithms	✗	✗
Matos *et al.* [17]	Forwarding neighbors	Distributed algorithm	✗	✓
Ozkasap *et al.* [18]	Forwarding neighbors	Fixed strategy	✗	✓
Esposito *et al.* [19]	Forwarding neighbors	Distributed algorithm	✓	✓
Altoaimy *et al.* [20]	Forwarding neighbors	Fixed strategy	✗	✗
Berendea *et al.* [25]	Node behavior	Specific rules	✗	✗
Cason *et al.* [26]	Message semantics	Specific rules	✓	✗
Xu *et al.* [27]	Dense connectivity	Specific rules	✓	✗
Saldamli *et al.* [28]	Network architecture	Specific rules	✓	✗
Ours	probability & neighbors	Reinforcement learning	✓	✓

Note: F_1

However, the above schemes [25], [26], [27], [28] are all customized improvements made for the blockchain, corresponding to specific consensus algorithms. Thus they cannot be directly applied to our scenario.

In summary, the forwarding probability and neighbors of the gossip protocol are two main factors that are difficult to balance. We give a comparison of the respective schemes in TABLE 1. It is worth noting that none of the above schemes [14], [15], [16], [17], [18], [19], [20], [25], [26], [27], [28] consider the neighbors' historical packet reception. The high speed dynamic change of the topology causes the neighbors around the UAV to change frequently. The existing works do not distinguish between the old and new neighbor nodes, which may send redundant message packets to the nodes that have already received the packets, leading to a decrease in the propagation efficiency and thus prolonging the propagation delay.

3 Preliminaries

In this section, we introduce the preliminaries of the gossip protocol, gossip about gossip, and multi-agent deep reinforcement learning involved in the proposal.

3.1 Gossip Protocol

The gossip protocol, also known as the epidemic protocol, is a well-known message propagation method [29]. As shown in Fig. 2, rumor mongering is a complex epidemic. The node state is either susceptible, infective or removed. The susceptible state means that the node does not know the update. The infective state means that the node knows the update and is actively sharing it with others. The removed state means that the node knows the update but is not spreading it. The rumor mongering is as follows. We plant a rumor with the red node, which becomes infective and propagates to others randomly. Every node hearing the rumor also starts propagating. Then, with probability 1/k, the infective node loses interest in sharing the rumor until it becomes removed.

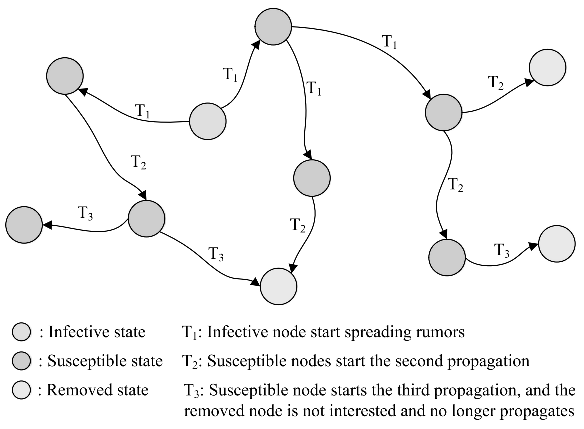

T₁
T₁
T₁
T₁
T₂
T₂
T₂
T₂
T₃
T₃
T₃
: Infective state
: Susceptible state
: Removed state
T₁: Infective node start spreading rumors
T₂: Susceptible nodes start the second propagation
T₃: Susceptible node starts the third propagation, and the removed node is not interested and no longer propagates

Fig. 2. The rumor mongering of gossip.

3.2 Gossip about gossip in Hashgraph

Hashgraph algorithm is an Asynchronous Byzantine Fault Tolerance (ABFT) consensus algorithm proposed by Baired *et al.* [30], which replaces the traditional blockchain data structure with a direct acyclic graph. Inspired by the piggyback propagation idea, Hashgraph propagated a packet reception record of each node when propagating the message packet. The above process is called gossip about gossip. As shown in Fig. 3, if node B has already received a message M from node A when node B propagates the message M to node C, it will tell node C about the action "node A propagates message M to node B".

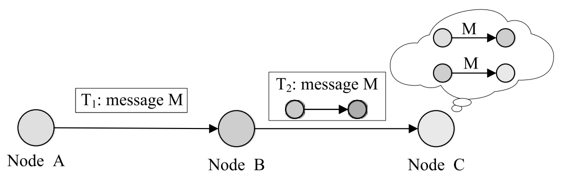


Fig. 3. The gossip about gossip.

3.3 Reinforcement Learning

3.3.1 Deep Reinforcement Learning

The problem studied by reinforcement learning is that of an intelligent agent i interacting with its environment, which is a sequential decision process [21]. When interacting with the environment, the agent i learns an optimal strategy through continuous trial-and-error exploration. Formally, the sequential decision process is modeled as a Markov Decision Process (MDP), which can be represented as a triplet $\{\mathcal{S}, \mathcal{A}, T, R\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the state transition function, and R is the reward function. Specifically, the agent i chooses an action $a(t) \in \mathcal{A}$ at a timeslot t based on the state $s(t) \in \mathcal{S}$. When the agent i performs an action $a(t)$, it moves to the next state $s(t+1)$ with a probability $T(s(t), a(t), s(t+1))$ and receives an immediate reward $r(t) = R(s(t), a(t), s(t+1))$ from the environment. Therefore, the action value function Q_μ for the timeslot t can be expressed as:

$$Q_\mu(s(t), a(t)) = \mathbb{E}_\mu[\sum_{k=t}^{T} \gamma^{k-1} \cdot r(k)] \tag{1}$$

where μ represents a deterministic policy in practice, $\gamma \in (0,1)$ is the discount factor. In summary, the goal of the agent i is to learn a strategy μ to maximize the action value function Q_μ, which can be expressed as:

$$\mu^* = \arg\max_a Q_\mu(\mathcal{S}, \mathcal{A}) \tag{2}$$

where μ^* is the optimal strategy.

Recently, we often utilize a function Q_ϕ to represent the approximate computation, $Q_\phi \approx Q_\mu$, and the function Q_ϕ is usually a neural network with parameters ϕ.

3.3.2 Deep Graph Netural Network

The multi-agent reinforcement learning of deep feature vector $\boldsymbol{h}_i^t$ by multi-layer perceptron (MLP) for low-dimensional input. The convolutional layer integrates the feature vector $\boldsymbol{h}_i^t$ in the local region and generates the latent feature vector $\boldsymbol{h}_i^{'t}$ and $\boldsymbol{h}_i^{''t}$. By stacking two layers, agent i can get two-hop information while only communicating with its neighbors.

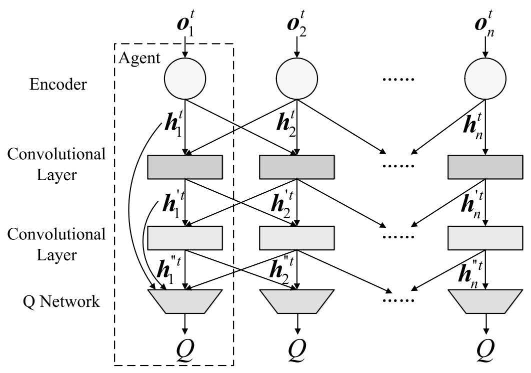


Fig. 4. The multi-agent reinforcement learning algorithm DGN.

4 PROBLEM FORMULATION

In this section, we first give the network model. Then, we give the problem definition. The notations frequently used are illustrated in TABLE 2.

TABLE 2
NOTATION DEFINITIONS

Notation	Meaning
t, T	Timeslot, The maximum timeslot of an episode
D_{com}	The communication range of UAV
m_i	The message received indicator
R_i	The current redundant message numbers
B_i	The current Bitgraph filling status
$\mathcal{HN}_i$	The neighbor list of the UAV i
p_g	The forwarding probability
$\mathcal{N}_{nb}$	The forwarding neighbors
$\boldsymbol{o}_i^t$	Local observation of UAV i at timeslot t
$\mathcal{O}_t$	The observation space at timeslot t
$\boldsymbol{a}_d$	The sub-action
r_t	The reward at timeslot t

4.1 Network Model

In our network model, we consider a scenario with a number of UAVs $\mathcal{N}_{uav} = \{1, \cdots, N\}$. These UAVs will be arranged to collect information in a specific area. As shown in Fig. 5, the network model of our proposal consists of UAVs. Due to the limited communication range D_{com} of UAVs, not all UAVs are within the communication range of each other. The gossip protocol is an efficient and scalable broadcast protocol, where each UAV constructs a locally maintained neighbor list $\mathcal{HN}_i$ through heartbeat packet messages before broadcasting. For example, UAV i has three neighbors. After the UAV i sends a message, the rest of the UAVs will relay the received packets to several of the surrounding neighbors with a certain probability. And this method can mitigate the impact of packet loss caused by unstable link characteristics in wireless networks. However, the settings of forwarding probability p_g and forwarding neighbors $\mathcal{N}_{nb}$ in the gossip protocol can significantly affect the effectiveness of the gossip protocol in different network conditions [12].

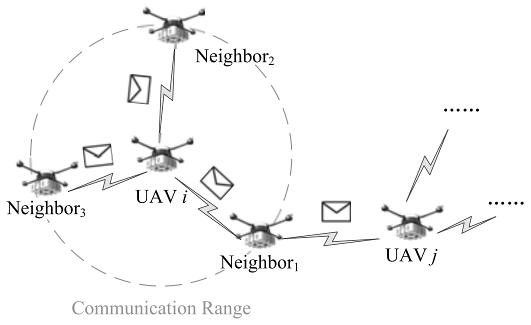


Fig. 5. Network model.

Therefore, we expect UAVs to adaptively adjust the propagation strategy of the gossip protocol in a dynamic environment by collaborating. Specifically, since the network topology and other network conditions are different when UAVs act as an edge pool, individual UAVs cannot be informed of the global network conditions due to their limited sensing ranges. We allow UAVs to sense the local network conditions on a larger scale through a collaborative approach and further decide their respective gossip protocol policy settings based on the sensing results.

4.2 Problem Definition

Intending to maximize the gossip efficiency, we optimize the forwarding probability p_g and forwarding neighbors $\mathcal{N}_{nb}$ policy of UAVs, subject to the maximum probability constraint, one-hop neighbors constraint and spacial constraints. Therefore, the propagation policy π determines the performance of the gossip and has to be optimized for message redundancy minimization and propagation delay minimization.

We introduce two global metrics to evaluate the performance of the Bitgraph-gossip protocol and illustrate the target problem we want to solve. According to [4], we evaluate the performance of the gossip strategy from the two indicators of message redundancy and propagation delay.

The first indicator is message redundancy, which represents the average number of redundant messages received by each UAV in the past t timeslots. For example, at any timeslot t, if the UAV receives previously received messages, these messages are counted. Therefore, the message redundancy can be expressed as:

$$MR_t = \frac{\sum_{i=1}^{N} w_t(i)}{}$$

and the denominator $N(N-1)(N-2)$ [1] is the total number of redundant messages received by all UAVs in extreme cases. Therefore, $MR_t \in [0,1]$ is established. We refer to the message redundancy of the last timeslot t as the final message redundancy, denoted as $MR_T = MR_{t|t=T}$, which is used to evaluate the performance of the UAV cluster in an episode.

The second indicator is the propagation delay, which represents the average time it takes for the message packets received by each UAV in the past t timeslots to arrive at the UAV. During broadcasting, each UAV $j \in \mathcal{N}_{uav}$ will broadcast the data packet, and the sending time is recorded as $start_j$, and when the other UAVs receive the data packet, the receiving time is recorded as end_j. Therefore, the propagation delay can be expressed as:

$$PL_t = \frac{\sum_{i=1}^{N} v_t(i)}{N(N-1)}, \tag{4}$$

where $v_t(i) = \sum_j end_j - start_j$ is the time taken by UAV i to receive other UAVs $j \in \mathcal{N}_{uav} \backslash \{i\}$ in timeslot t, and N is the number of UAVs $|\mathcal{N}_{uav}|$. We refer to the message redundancy of the last timeslot T as the final propagation delay, denoted as $PL_T = PL_{t|t=T}$, which is used to evaluate the performance of the UAV cluster in an episode.

Therefore, the objective of the proposed propagation algorithm design is to minimize message redundancy and propagation delay.

Further, we combine these two metrics and define the overall goal as the redundancy-latency (RL) score, expressed as:

$$RL_t = \frac{1}{MR_t \times PL_t} \tag{5}$$

Hence, the objective function of distributed propagation policy optimization problem for UAVs can be formulated as follows:

$$\max RL = \sum_{t=0}^{T} RL_t \tag{6}$$

$$s.t. x_{\min} \le x_i^t \le x_{\max}, \forall i \in \mathcal{N}_{uav} \tag{7}$$

$$y_{\min} \le y_i^t \le y_{\max}, \forall i \in \mathcal{N}_{uav} \tag{8}$$

$$(x_i^t, y_i^t) \ne (x_j^t, y_j^t), i, j \in \mathcal{N}_{uav}, \forall t \tag{9}$$

$$p_g^t \in \{0, 10\%, 20\%, \ldots, 100\%\}, \forall t, \forall i \in \mathcal{N}_{uav} \tag{10}$$

$$\mathcal{N}_{nb}^t \subseteq \mathcal{N}_{uav}, \forall t, \forall i \in \mathcal{N}_{uav} \tag{11}$$

where (7) and (8) indicate the constraints for the UAVs' position, which has to be in the airspace above the MEC needed within a fixed height. Constrain (9) is invoked to ensure that there is no collision between UAVs. Constrain (10) specifies the UAV forwarding probability of between 0 and 100%, with 11 grades. Constrain (11) indicates the UAVs forwarding neighbors are some of the one-hop neighbors.

However, it is challenging to achieve the above target. This is because the forwarding probability and neighbors of the gossip protocol are two contradictive factors that are difficult to balance. Besides, it is a control problem that the target problem cannot solve via the conventional dynamic programming method with an unknown environment, which is a model based approach. To overcome the above issue, we choose to use deep reinforcement learning (DRL) to find optimal solutions, which can learn policy from interactive experience without requiring prior information about the environment. However, since the challenge mentioned above, UAVs' rewards are sparse and affected by the actions of many other UAVs. Hence, the objective function of the above problem can be formulated as follows:

$$\pi^* = \arg\max_{\pi} RL \tag{12}$$

where T is the last timeslot in the episode, π is the UAV policy of forwarding probability and neighbors. The solution to this problem is presented in the next section.

5 Our Proposal: Intelligent Adaptive Gossip-based Broadcast Protocol

In this section, we give the design of our intelligent adaptive gossip-based broadcast protocol. We first give the overview of our proposal. Then, we give more details.

5.1 Overview

To address the defined problem, we first introduce a data structure, Bitgraph, to record the packet reception of UAVs. Inspired by the gossip about gossip from Hashgraph, our Bitgraph can update during gossip propagation. We give the updating instruction with a six-step demo, including the stopping and consistency check rules. Next, we formulate the gossip broadcasting as a partially observable Markov decision process (POMDP). In each decision cycle, UAVs choose an action and get a reward according to the partial observation from the environment. It is worth noting that a well-designed reward function is presented to guide UAVs in learning how to gossip based on the Bitgraph. We extract the change situation between two rounds as an intrinsic reward. Hence, we can resolve the sparse reward problem. Then, considering there are two factors as actions need to decide simultaneously, we propose our reinforcement learning algorithm BDGN, consisting of a branching structure and DGN. Each network branch deals with a sub-action. DGN convolves the features of adjacent UAVs, allowing UAVs to explore strategies in the environment until they learn a good model. The overview of our proposal is shown in Fig. 6.

1. In extreme cases, any two UAVs are connected. If UAV_1 is the source. Other UAVs will forward the message to the rest of the UAVs (except source UAV_1) after receiving the message. From the perspective of UAV_2, it will receive N-2 redundant messages. Likewise, UAV_2 to UAV_N will receive N-2 redundant messages. Thus, in the case of UAV_1 as the source, the number of redundant messages is (N-1)(N-2). We assume that each node is a message source, so the total number of message redundancies is N(N-1)(N-2).

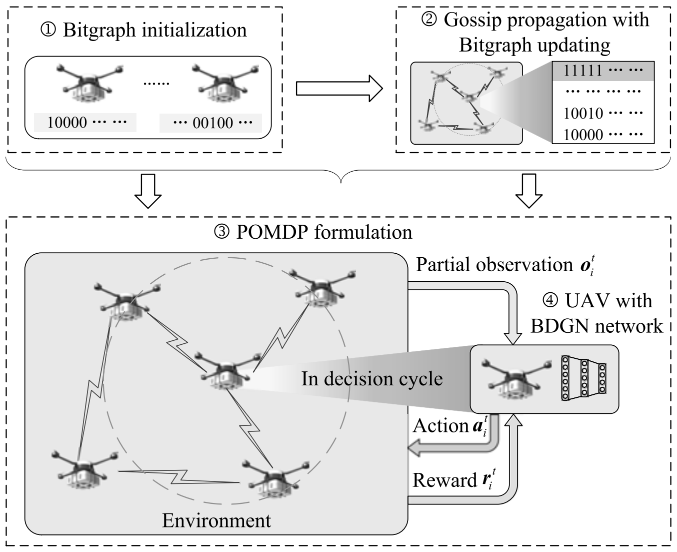


Fig. 6. The overview of our proposal.

deployment, the string length is the number of members squared, and the string is partitioned and subdivided in the logical order of the members. As shown in Fig. 7, for example, with three UAVs, the string length is 9, each UAV has a corresponding zone in a logical order, and each zone has corresponding bits in logical order. In UAV A's partition, the first 1 means that UAV A sent its message packet. The second 1 means that UAV A received the message packet sent by UAV B, and the third 0 means that UAV A received no message packet from UAV C. In the partition of UAV B, the first and third 0 means that UAV B did not receive the message packets sent by the corresponding bit UAV A and C. The second 1 means that UAV B sent its message packet outward. In the partition of UAV C, the first and second 1 mean that UAV C received the message packets sent by UAV A and B, and the last 1 means that UAV C sent its message packet outward.

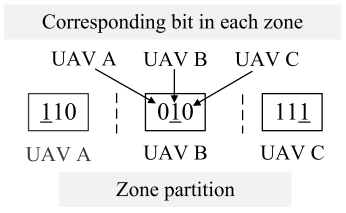


Fig. 7. Diagram of Bitgraph from UAV A's viewpoint, this Bitgraph belongs to UAV A, and the underlined bit represents whether there is a message packet sent from the corresponding UAV.

5.3 Gossip propagation with Bitgraph updating

We give a demo of gossip propagation with Bitgraph updating. Before the formal description, we introduce two rules.

The Bitgraph-gossip protocol stopping rule is: When the UAV's Bitgraph is full of 1, the UAV stops gossip and Bitgraph updating after the last message forwarding.

The Bitgraph-gossip protocol consistency check rule is: When the UAV finds that its Bitgraph is not yet filled with 1 (i.e., when it has not yet received all the message packets) without listening to the messages sent to itself, the UAV can communicate with the corresponding neighbor UAV and request retransmission of the missing message packets according to the Bitgraph.

For brevity, we introduce Bitgraph-gossip propagation with three UAVs as an example and specify that the UAVs choose only one neighbor UAV for message forwarding when performing gossip propagation. As shown in Fig. 8, after the last round of message broadcast, the Bitgraph of the UAVs first performs a zeroing operation. Then, the UAVs generate message packets separately, update the 0 to 1 within the Bitgraph, and propagate the message packets and their Bitgraphs through the gossip protocol. We divide the gossip propagation process into six steps.

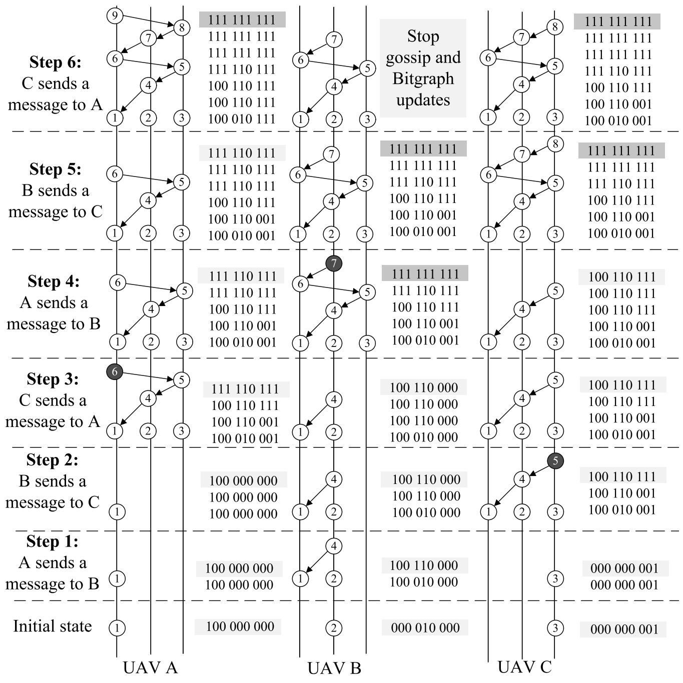

Step 6: C sends a message to A
Step 5: B sends a message to C
Step 4: A sends a message to B
Step 3: C sends a message to A
Step 2: B sends a message to C
Step 1: A sends a message to B
Initial state
Stop gossip and Bitgraph updates
UAV A
UAV B
UAV C
111 111 111
111 110 111
100 110 111
100 110 001
100 010 111
100 010 001
100 000 000
100 110 000
100 010 000
000 010 000
000 000 001

Fig. 8. Diagram of message broadcast under Bitgraph (gray shaded binary string represents the latest Bitgraph of the UAV at the moment, red shaded means the UAV's Bitgraph is filled in this step, a red circle means the UAV has received all the message packets actually in this step).

Step 1: When UAV B receives a message from UAV A, UAV B updates UAV A's partition in Bitgraph to 100, which means that UAV A generated and sent its message packet. In the same way, UAV B updates UAV B's partition to 110, which means that UAV B generated and sent its message packet and received UAV A's message packet.

Step 2: When UAV C receives a message from UAV B, UAV C updates UAV A's partition in Bitgraph to 100, which means that UAV A generated and sent its own message packet. Similarly, UAV C updates UAV B's partition to 110, which means that UAV B generated and sent its own message packet and received UAV A's message packet, and it updates UAV C's partition to 111, which mean

UAV B has not received UAV C's message packet, and it updates UAV C's partition to 111, which means that UAV C has received all the message packets.

Step 4: UAV A sends a message packet to UAV B. When UAV B receives the message from UAV A, UAV B updates the Bitgraph to 111 111 111, at which point UAV B receives all the message packets and assumes that the rest of the UAVs have also received the message packets. UAV B will forward the message packet one last time, and after sending the message, UAV B will stop gossip and the Bitgraph updating.

Step 5: UAV B sends a message packet to UAV C. UAV C receives the message and updates the Bitgraph to 111 111 111, at which point UAV C receives the message packet and assumes that the rest of the UAVs have also received the message packet. UAV C will forward the message packet one last time, and after sending the message, UAV C will stop gossip and Bitgraph update. Similarly, UAV C will forward the message packet one last time.

Step 6: UAV C sends a message packet to UAV A. UAV A updates Bitgraph to 111 111 111, at which point UAV A receives the message packet and assumes that the rest of the UAVs have also received the message packet. Similarly, UAV A will forward the message packet one last time and then go into silence.

Due to the unstable link, there will be packet loss in gossip messages. If UAV A does not receive the message packet from UAV C in step 6, UAV A's Bitgraph is still 111 110 111. In this case, UAV A thinks that UAV B has not yet received all the message packets, so message synchronization will be performed between UAV A and B. Specifically, UAV A requests the Bitgraph of UAV B. Since the received Bitgraph is 111 111 111, which means that UAV B has received all the message packets, UAV A updates its Bitgraph to 111 111 111. At this point, the consistency check is over, and all three UAVs have received all the message packets.

5.4 POMDP Formulation

The gossip broadcasting of UAVs can be formulated as a POMDP since the UAVs make sequential decisions with a partial observation limited by communication range. The POMDP can be defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, T, R, O, b\}$. However, due to the whole environment being unknown for UAVs, state transition function T, observation function O, and belief b are unknown. We give the rest of the POMDP as follows.

5.4.1 State Space

The state space $\mathcal{S}$ in our system is the all UAVs' local state. The mathematical expression of the state space in the t-th timeslot can be expressed as:

$$\boldsymbol{s}^t = (m_i, R_i, B_i) \tag{13}$$

where $i \in \mathcal{N}_{uav}$, m_i represents whether the UAV i receives the message packet, R_i represents the current message redundancy of the UAV i, and B_i represents the current Bitgraph filling status of the UAV i. Hence, the state space is $\mathcal{S} = \{\boldsymbol{s}^t\}$, where $i \in \mathcal{N}_{uav}$.

5.4.2 Observation Space

For UAV i in the t-th timeslot, it perceives a local observation $\boldsymbol{o}_i^t$ from the environment. We describe in Section 4.1 that the communication range to a UAV i is D_{com}, so the observation result is the case of other UAVs within the range D_{com} centered on UAV i. Therefore, the observation is:

$$\boldsymbol{o}_i^t = (m_j, R_j, B_j) \tag{14}$$

where $j \in (\mathcal{HN}_i \bigcup i)$. Hence, the observation space is $\mathcal{O} = \{\boldsymbol{o}_i^t\}$, where $i \in \mathcal{N}_{uav}$.

5.4.3 Action Space

In the process of broadcasting through the gossip protocol, the UAV i can decide on its probability p_g of forwarding packets and forwarding neighbors $\mathcal{N}_{nb}$. In terms of forwarding probability, the UAV i needs to decide with a proper probability to forward the received packet. We divide the probability into multiple grades, and the action space of forwarding packet probability is

$$\mathcal{A}_1 = \{0, 10\%, 20\%, \cdots, 100\%\} \tag{15}$$

In determining the forwarding neighbor, as described in Section 4.1, the UAV i maintains a neighbor list $\mathcal{HN}_i$, and it needs to decide which neighbor UAV to forward the message to, and the action space of the forwarding neighbor is $\mathcal{A}_2 = \mathcal{N}_{nb} \subseteq \mathcal{HN}_i$.

5.4.4 Reward Function

As described in subsection 4.2, the goal of UAVs is to maximize the RL score. Therefore, a straightforward idea is to directly use this score as a reward [32], ie $r_i^t = RL_t$. However, since our system is modeled as an ad hoc network of UAVs with partially observable capabilities, global metrics are not available during the training process of multi-agent reinforcement learning. Furthermore, even though we can obtain RL by means of centralized training, due to partially observable constraints, distributed models have information errors in predicting global RL, which may lead to extremely unstable training. Therefore, we want to design a reward function that only depends on partially observable information. We hope that such a reward function can encourage UAVs to achieve better RL scores, which represent lower message redundancy MR_t and lower propagation delays PL_t. Since our reinforcement learning scheme is to obtain the maximum cumulative reward:

$$R_i^T = \sum\nolimits_{i=1}^{N} \sum\nolimits_{t=1}^{T} \gamma^{t-1} r_i^t \tag{16}$$

Ideally, we want the proposed reward function to be positively related to the RL score, i.e.

$$\sum\nolimits_{i=1}^{N} \sum\nolimits_{t=1}^{T} \gamma^{T-t} r_i^t \propto RL_T = \frac{1}{MR_T \times PL_T} \tag{17}$$

where γ is the discount factor.

According to the above analysis, we want to use propagation delay and message redundancy as the design basis of the reward function. However, the propagation delay reward $r_{bitclock}$

The idea is: When people forward news in real life, they will lose the desire to spread the widely circulated news. And when spreading the news, they tend to spread to people who don't know it.

Therefore, we design an intrinsic reward mechanism where at each timeslot t, the UAV checks the local message redundancy and the filling situation of the Bitgraph. For message redundancy, we divide the number of redundant messages by the number of UAVs in one-hop communication and then take the reciprocal of the calculation result as a reward r_t^{igroup}. For the filling situation of Bitgraph, we calculate the filling percentage of 1, and then subtract the filling percentage of timeslot t from the filling percentage of the previous timeslot t-1. If the subtraction result is 0, the reward $r_{bitclock}$ is recorded as -1; otherwise, The $r_{bitclock}$ reward is recorded as the result of the subtraction multiplied by 100.

To sum up, the reward function we propose is mainly composed of three parts, namely propagation delay reward $r_{bitclock}$, message redundancy reward r_t^{igroup} and Bitgraph filling percentage reward $r_t^{bitgraph}$.

Specifically, the propagation delay reward $r_{bitclock}$ in the first part comes from our Bitgraph. Since Bitgraph will be cleared before broadcasting, when it is cleared, the UAV i will turn on the timing clock locally. With the progress of broadcasting, when the Bitgraph is filled with 1, it means that the UAV i thinks that everyone has received the message packet, and it stops the timing clock locally. We get the timing results as t_i. And the delay reward for this broadcasting is:

$$r_{bitclock} = \frac{1}{t_i} \tag{18}$$

The second part is the message redundancy reward r_t^{igroup}, which is defined as the average message redundancy of UAV i and its neighbors, which aims to encourage adjacent UAVs to cooperate. And the message redundancy reward is:

$$r_t^{igroup} = 1 \div \frac{n_t^{count}}{n_{onehop}} = \frac{n_{onehop}}{n_t^{count}} \tag{19}$$

where n_{onehop} represents the number of UAV groups containing UAV i and its neighbors within one hop, and n_t^{count}represents the number of redundant messages recorded by these UAVs in the timeslot t.

The third part is the local Bitgraph filling percentage reward

$$r_t^{bitgraph} = \begin{cases} 100 \times (b_t - b_{t-1}) \\ 0, b_t - b_{t-1} = 0 \end{cases} \tag{20}$$

where b_t is the filling percentage of the bitgraph in the timeslot t, which is the number of 1 in the bitgraph divided by the bitgraph string length, and b_{t-1} is the filling percentage of the bitgraph in the previous timeslot $t-1$.

In summary, our heuristic reward function can be written as:

$$\begin{aligned} r_i = &\sum\nolimits_{t=1}^{T-1} \gamma^{t-1}(0.01 \times r_t^{igroup} + 0.1 \times r_t^{bitgraph}) \\ &+\gamma^T(0.01 \times r_t^{igroup} + 0.1 \times r_t^{bitgraph} + 0.1 \times r_{bitclock}) \end{aligned} \tag{21}$$

Among them, γ is the discount factor. Since the most important evaluation index is the propagation delay, the proportion of future rewards $r_{bitclock}$ is required to be large, and the discount factor $\gamma \in [0, 1]$ should be close to 1.

5.5 Branching Deep Graph Network

Now, we introduce the designed branching deep graph network (BDGN). BDGN is an improved deep learning reinforcement algorithm suitable for the gossip propagation protocol of UAVs. It is designed to process two sub-actions simultaneously under the propagation strategy of the gossip protocol. In this scheme, all UAVs communicate through the ad-hoc network and learn the gossip protocol policy in a distributed manner. Next, we introduce the detailed composition of the BDGN.

The design inspiration for BDGN comes from Branching Dueling Q-Network (BDQ) [23] and Deep Graph Network (DGN) reinforcement learning [31]. Both are improvements on DQN, in which BDQ designs a shared decision module based on DQN, which is followed by several network branches to deal with different sub-actions. DGN convolves the features of adjacent UAVs, allowing the agent to extract potential features from gradually increasing receptive fields to learn cooperative strategies. Compared to the conventional multi-agent deep reinforcement learning-based algorithm, we innovatively combine graph attention mechanism and branching structure network. Specifically, we have a shared decision module followed by two network branches corresponding to forwarding probability and neighbors. Besides, the shared decision module comprises two graph attention layers designed specifically for processing graph-structured data and considering node relationships. It is a suitable choice for capturing complex dependencies in the graph [31], [33].

In this subsection, we will introduce the specific composition of BDGN, as shown in Fig. 9. From top to bottom in Fig. 9(a), the first is the observation embedding layer, the second is the branch graph attention network (GAT) layer, and the third is the Q-value prediction layer. Fig.9(b) gives the specific structure of the graph attention network layer. In Fig.9(c), the importance of the neighborhood around the distribution UAV 1 is different due to the spread of the message, i.e., the highest score for UAV 2 and the lowest score for UAV 3. With the second GAT layer, the cooperative scope of UAV 1 expands to 12 UAVs. Such additional information helps UAV 1 pay more attention to the direction of the propagation of the message that needs to be received.

5.5.1 Observation Embedding

First, given the local original observation data, that is, whether the UAV i receives the message and the message redundancy of other UAVs within the communication range. We embed observation data into a latent vector through a multi-layer perceptron:

$$\boldsymbol{e}_i = Embed(\boldsymbol{o}_i^t) = \sigma(\boldsymbol{o}_i^t \boldsymbol{W}_e + \boldsymbol{b}_e) \tag{22}$$

where $\boldsymbol{o}_i^t$ is the observation of UAV i at timeslot t, detailed composition of $\boldsymbol{o}^t$

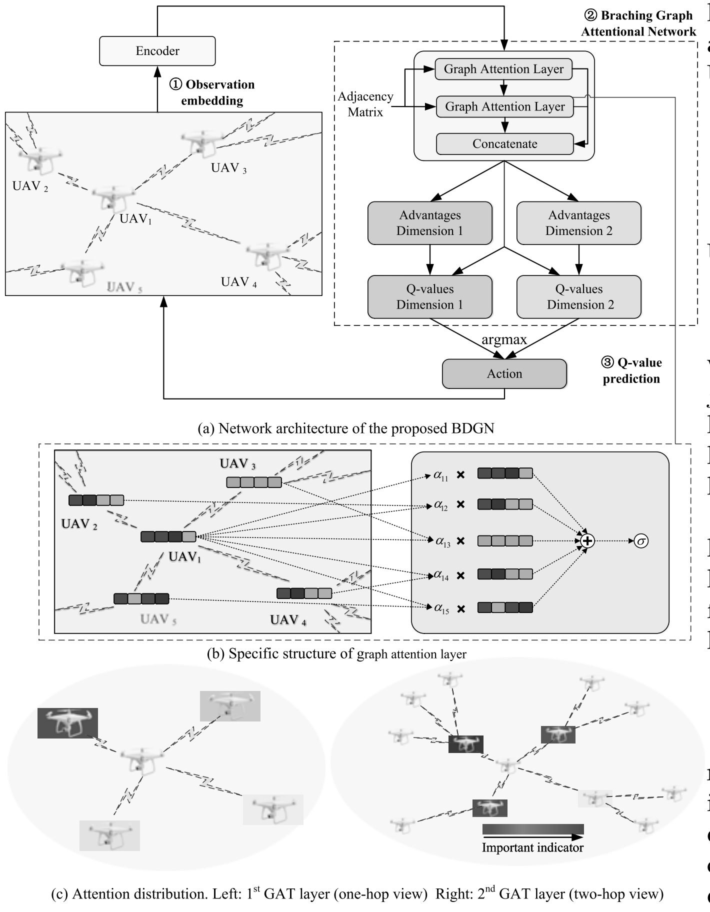


Fig. 9. The proposed BDGN.

e_i represents the current observation situation of UAV i. Before input to BDGN, the reason for using MLP to process observation instead of using raw one-dimensional data is that MLP can transform observation into a more compact, higher-dimensional representation. In this way, MLP can help the network to learn the important features in the data, thus improving the efficiency and performance of subsequent tasks.

Then, the embedding of each UAV i will be transmitted through the edges between UAVs so that each UAV can obtain information about itself and its surrounding neighbors. For convenience, we denote the UAV i and its one-hop neighbor set as

$$\mathcal{NB}_i = \{j|\forall j s.t. AM(i,j)=1\} \tag{23}$$

where AM is the adjacency matrix of the UAVs. Correspondingly, we denote the observation space $\mathcal{O}_i$ and embedding space $\mathcal{E}_i$ of all UAVs in the set $\mathcal{NB}_i$ as follows:

$$\begin{aligned}\mathcal{O}_i &= \{\boldsymbol{o}_j|\forall j \in \mathcal{NB}_i\}\\ \mathcal{E}_i &= \{\boldsymbol{e}_j|\forall j \in \mathcal{NB}_i\}\end{aligned} \tag{24}$$

5.5.2 Branch Graph Attention Network

In a neighbor set $\mathcal{NB}_i$, the importance of each neighbor UAV to UAV i should be different. For instance, when UAV i finds that neighbors in a certain direction have low message redundancy, these UAVs should be more important to i. Meanwhile, we adopt the graph attention as the convolution kernel to process the graph data and utilize the graph attention mechanism [34] to decide the importance of each UAV as shown in Fig.9(b), calculated as:

$$\alpha_{ij} = \frac{\exp(\tau \cdot \boldsymbol{W}_Q \boldsymbol{e}_i \cdot (\boldsymbol{W}_K \boldsymbol{e}_j)^T)}{\sum_{k\in\mathcal{NB}_i} \exp(\tau \cdot \boldsymbol{W}_Q \boldsymbol{e}_i \cdot (\boldsymbol{W}_K \boldsymbol{e}_k)^T)} \tag{25}$$

Further, the aggregation of neighbors' information of UAV i can be expressed as:

$$\boldsymbol{g}_i^1 = \sigma((\sum\nolimits_{j\in\mathcal{NB}_i} \alpha_{ij} \cdot \boldsymbol{W}_V e_j) \cdot \boldsymbol{W}_\theta + \boldsymbol{b}_\theta) \tag{26}$$

where α_{ij} is the attention weight of the importance of UAV j to UAV i. τ is a scaling factor. $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ are the learning matrices that map the embeddings $\boldsymbol{e}_i$ to query, key, and value vectors respectively [34]. $\boldsymbol{W}_\theta$, $\boldsymbol{b}_\theta$ are also the learning matrices.

The above is the computation process of graph attention layer (GAT) [35]. As shown in Fig. 9(a), we stack two GAT layers together, which can provide a two-hop perception field for each UAV. Therefore, we have the output of second layers can be expressed as:

$$\boldsymbol{g}_i^2 = GAT(\boldsymbol{g}_i^1) \tag{27}$$

The UAV i can obtain the embedded information of its neighbors in the first layer from the second layer GAT, so it obtains the information of all UAVs within two hops of UAV i. Note that in this process, UAV i only needs to communicate with UAVs within one hop. Specifically, the outputs of each layer of GAT are concatenated as the final output:

$$\boldsymbol{g}_i = concatennate(\boldsymbol{g}_i^1||\boldsymbol{g}_i^2) \tag{28}$$

When concatenating GAT 1 with GAT 2, it is worth noting that GAT 1 focuses on capturing local neighborhood relationships and node-level information, while GAT 2 captures more global patterns and higher-level dependencies.

5.5.3 Q-value Prediction

At the same time, in order to enable the proposed network to cope with the two sub-actions in the gossip strategy, BDGN inputs the outputs in the double-layer GAT to the two action branches respectively, where the advantage and state values are combined through a special aggregation layer on the action branch, and is computed with an independent state value to produce an estimate of the distributed action value. When the Q value on each branch is in state $\boldsymbol{o}_i^t$ and sub-action $\boldsymbol{a}_d^t \in \mathcal{A}_d$, $Q_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t)$ can be calculated through the common state value function $V(\boldsymbol{o}_i^t)$ and the corresponding sub-action advantage $\boldsymbol{A}_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t)$, where d is the label of the branch. The common state value function $V(\boldsymbol{o}_i^t)$ and the corresponding sub-action advantage $A_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t)$ can be expressed as:

$$V(\boldsymbol{o}_i^t) = \sigma(\boldsymbol{g}_i \boldsymbol{W}_\alpha + \boldsymbol{b}_\alpha) \tag{29}$$

$$A_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t) = \sigma(\boldsymbol{g}_i \boldsymbol{W}_{\beta_d} + \boldsymbol{b}_{\beta_d}) \tag{30}$$

where $\boldsymbol{W}_\alpha$, $\boldsymbol{b}_\alpha$, $\boldsymbol{W}_{\beta_d}$, $\boldsymbol{b}_{\beta_d}$

The calculation of $Q_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t)$ can be expressed as:

$$Q_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t) = V(\boldsymbol{o}_i^t) + (A_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t) - \underset{\boldsymbol{a}_d^{'t} \in \mathcal{A}_d}{\arg\max} A_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^{'t})) \tag{31}$$

To sum up, the process of the UAV interacting with the environment can be summarized as follows. At each timeslot, each UAV i acquires the observation space $\boldsymbol{o}_i^t$, and inputs it into the network to get action. The UAV implement action and gets a reward r_i^t and a new observation space $\boldsymbol{o}_i^{t+1}$ after execution.

Referring to the double DQN (DDQN) [36], we use a network that updates parameters to select actions and the fixed target network to calculate the value. The target network is a copy of the BDGN network whose parameters are updated every few iterations by copying the parameters from the trained model. We update the model by minimizing the expected value of the averaged temporal difference error. When using the temporal difference approach, we need to identify the temporal difference target so as to approximate the optimal policy step by step by updating the estimated value of the Q function. Refer to the [23], the temporal difference target y_d can be represented as follows:

$$y_d = r_i^t + \gamma \frac{1}{2} \sum_{d=1}^{2} Q_d^-(\boldsymbol{o}_i^{t+1}, \underset{\boldsymbol{a}_d^{'t} \in \mathcal{A}_d}{\arg\max} Q_d(\boldsymbol{o}_i^{t+1}, \boldsymbol{a}_d^{'t})) \tag{32}$$

where Q_d^- denoting the branch d of the target network Q^-.

Based on the above equation, the loss function is calculated as:

$$L = \frac{1}{2N} \sum_{i=1}^{N} \sum_{d=1}^{2} (y_d - Q_d(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t))^2 \tag{33}$$

Once the loss L is obtained, we can use gradient descent to update the parameters of the BDGN network iteratively until convergence.

The proposed BDGN is summarized in Algorithm 1. Since the UAVs are equal within the cluster, we only train a generic model, as shown in Algorithm 1. Before the UAV cluster is deployed as an edge pool, we initialize a BDGN network with parameter θ. Similarly, we make a copy for the target network and set an experience replay pool $\mathcal{D} = \varnothing$ and a maximum number of iterative rounds N_{ep}. After that, we randomize the locations of the UAVs. At first, let them propagate according to the ε-greedy strategy and obtain the corresponding reward r_i^t and observation space $\boldsymbol{o}_i^{t+1}$, and store these experiences $(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t, r_i^t, \boldsymbol{o}_i^{t+1})$ in the replay pool. Then, we extract a small batch of experiences, let the model calculate the Q value according to Eq. (30), and calculate the corresponding loss according to Eq. (31) and Eq. (32). Next, based on the loss and gradient descent method to update the parameters of the BDGN network. Finally, we update the target network only at a fixed training interval for training stability.

5.6 Computational Complexity Analysis

In this subsection, we analyze the computational complexity of the proposed BDGN algorithm. Referring to work [37], we measure the computational complexity in terms of both space complexity and time complexity.

Algorithm 1 BDGN Algorithm

```
Input: Initialize a BDGN network Q of parameters $\theta$ and
    target network $Q^-$ with parameters $\theta' \leftarrow \theta$; replay
    buffer $\mathcal{D} = \varnothing$; maximum number of traning episodes
    $N_{ep}$.
Output: Trained parameters $\theta$.
 1: for $n_{ep} = 1$ to $N_{ep}$ do
 2:   Randomly reassign the position f UAVs.
 3:   Set time step $t = 0$.
 4:   while Some drones with unfilled Bitgraphs do
 5:     time step $t++$.
 6:     for agent $\forall i \in \mathcal{N}_{uav}$ do
 7:       Obtain the observation $\boldsymbol{o}_i^t$.
 8:       Select the action $\boldsymbol{a}_d^t$ with $\varepsilon$ greedy strategy.
 9:       Execute the action $\boldsymbol{a}_d^t$, then obtain the corre-
          sponding reward $r_i^t$ and next observation $\boldsymbol{o}_i^{t+1}$.
10:       Store experience $(\boldsymbol{o}_i^t, \boldsymbol{a}_d^t, r_i^t, \boldsymbol{o}_i^{t+1})$ in butter $\mathcal{D}$.
11:     end for
12:     Randomly sample a small batch of tuples from the
        butter $\mathcal{D}$.
13:     Calculate the Q value based on (30).
14:     Calculate the target $y_d$ and loss $L$ based on (31) and
        (32).
15:     Train parameter $\theta$ with a gradient descent step L.
16:     Update BDGN network.
17:     if $t$ mod traning-interval = 0 then
18:       Update target network $Q^-$ by: $\theta' \leftarrow \theta$.
19:     end if
20:   end while
21: end for
```

5.6.1 Space Complexity

For the sake of concise and understandable analysis, we assume there are H hidden layers, each layer has c neurons, the input dimension of the observation space is s and the output action dimension is l. Then the size of the weight matrices and bias vectors in each component of BDGN is: 1)Observation embedding layer according to the formula (21): $sc + c$; 2)Two graph attention layers according to the formula (24), (25) and (26): $2(3\,|\mathcal{NB}_i| \cdot c^2 + (c^2 + c))H$; 3)Q-value prediction layer according to the formula (28), (29) and (30): $3(cl + l)$. Typically, the size of the hidden layer c is much larger than the number of layers H, the action space l and is similar to the input dimension s. Besides, $|\mathcal{NB}_i|$ is a constant. Hence, we get the space complexity of BDGN is approximately equal to $\mathrm{O}(c^2 H)$.

5.6.2 Time Complexity

In time complexity analysis, according to the convention, addition is not so important, and we count the number of multiplications. Therefore, the time complexity of BDGN is: 1) Observation embedding layer: sc; 2)Two graph attention layers: $2(|\mathcal{NB}_i| \cdot c^2 + c^2)H$; 3)Q-value prediction layer: $3cl$. Typically, the size of the hidden layer c is much larger than the number of layers H and action space l and is similar to the input dimension s. Besides, $|\mathcal{NB}_i$

complexity of action selection is $O(c^2H)$. And we get the time complexity of BDGN during the training process is approximately equal to $O(BN_{ep}c^2H)$.

6 Experiments

6.1 Settings

Since it is difficult to carry out experiments on a real UAVs network, as shown in Fig. 10, we adopt the discrete network event simulation simulator NS-3 version 3.25 as the simulation environment of the UAVs network, and the reinforcement learning framework PyTorch 1.7.1 is used to implement specific reinforcement learning algorithms. The two-part are coupled using Open AI's GYM 0.25 code framework. Then we train in a computer with Intel Xeon E-2176M@2.7GHz CPU and NVIDIA Tesla T4 GPU. The code of the whole prototype system has been released as open-source at the following link: https://github.com/lzwgiter/Intelligent-Adaptive-Gossip-based-Broadcast-Protocol. This will help academia and industries to verify our work and, in general, boost research toward intelligent communication in UAV-MEC.

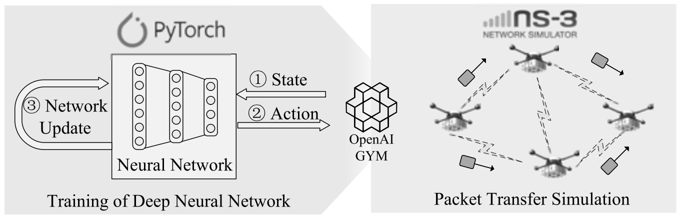


Fig. 10. Architecture of the simulation environment.

In order to simulate the broadcasting scenario of the UAVs, we add gossip protocol in the source code of NS-3 and use IEEE 802.11b to simulate the communication link between UAVs in which the log-distance path loss model is adopted, and the specific simulation parameters are shown in TABLE 3.

TABLE 3
NS-3 SIMULATION ENVIRONMENT PARAMETER SETTINGS

Parameters	Value
Simulation area	500*500 m
Number of UAVs	25, 50, 75, 100, 125 (default 25)
Shared data packet size	64 KB
Propagation loss model	ns3::LogDistancePropagationDelayModel
UAV speed	50-100 m/s
UAV total energy	550 KJ
Protocol	Flood/Gossip/Ours

6.2 Compared Methods

We compare the proposed BDGN with the latest deep reinforcement learning algorithms: DQN, CommNet, and DGN. A brief introduction of these algorithms is as follows:

1) DQN. DQN is the most widely used reinforcement learning algorithm, mainly used for learning in a single-agent environment [38]. Later some works use multiple Q-networks to represent multiple agents and introduce them into a multi-agent environment [39].

2) CommNet. CommNet is a classic multi-agent reinforcement learning algorithm [40]. In view of the fact that it is difficult for a single agent to observe the global state of the system in the real environment, the agents improve the learning efficiency and effect by learning and communicating.

3) DGN. DGN is a recently proposed multi-agent reinforcement learning algorithm that uses a convolutional graph network as a communication module to improve communication efficiency and collaboration effects [31]. However, it does not consider the case of multiple sub-actions. We compare it to show the importance of shared decision making modules.

Meanwhile, we compare the proposed message broadcast approach with flood based, gossip based, forwarding probability decision based, and forwarding neighbor decision based broadcast approaches.

1) Blinded Flood based broadcasting approach (BFBA). Multiple source UAVs send message packets to all their neighbors. When receiving a new message packet, a relay UAV sends it to all its neighbors. All UAVs forward a message packet and receive some packets once, called a round. The algorithm completes convergence when all the UAVs in the network receive message packets from multiple source UAVs.

2) Random gossip based broadcasting approach (RGBA). Instead of forwarding data randomly to all its neighbors, multiple source UAVs forward message packets with a 60% probability to 3 randomly selected neighbors. The relay UAV forwards the message packets with a 60% probability to three randomly selected neighbor UAVs. The convergence requirement of the algorithm is the same as BFBA.

3) Changeable forwarding probability based broadcasting approach (CFPBA) [15]. In this approach, the decision idea of the relay UAVs for forwarding probability when receiving a message is that the more times a UAV receives a redundant message, the lower the probability of continuing to relay this redundant message.

4) Suitable forwarding neighbor based broadcasting approach (SFNBA) [20]. This approach scores the neighbor UAVs based on the distance relationship between UAVs and selects the most suitable forwarding neighbor UAV. The basic idea is that the closer the neighbors are, the higher the score.

6.3 Neural Network Convergence and Reward Function Effectiveness

6.3.1 Neural Network Convergence Comparison

As shown in Fig. 11, we demonstrate the convergence of all tested DRL methods by the cumulative trend of the reward function during training. We note that our proposed BDGN converges faster and behaves more stably than other algorithms. This is because a shared decision making module is added to BDGN, which coordinates the decision to learn the interaction between various agents and uses the latent features generated by the convolution layer to learn cooperation. Thus it has a better effect in the scenario where UAVs cooperate in spreading the message.

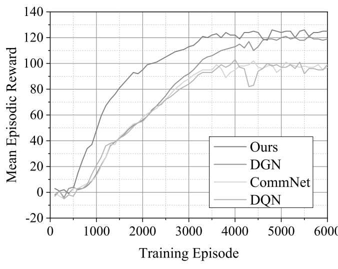


Fig. 11. Cumulative curve of reward function in training phase.

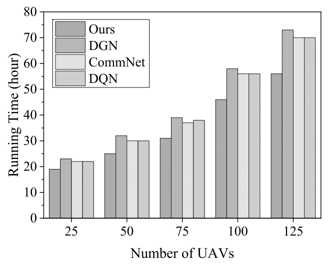


Fig. 12. Time required for model training convergence.

To further test the convergence of the neural network, we train the neural network by varying the number of UAVs. We set the number of UAVs to 25, 50, 75, 100, and 125 and counted the time it took for the model to converge. As shown in Fig. 12, with the number of UAVs increasing, the time required for model convergence also increases. The reason is that the increase in the number of UAVs will increase the complexity of the problem. The number of messages forwarded by each UAV will also increase accordingly to spread the message to the entire network. Our model can achieve the fastest convergence speed regardless of the number of UAVs because our model adds a shared decision module compared to DGN, CommNet, and DQN, which speeds up the convergence of the model speed. The model convergence speed of DGN is the slowest, which is in line with our expectations because DGN adds an attention mechanism compared to CommNet and DQN, which increases the difficulty of model convergence.

6.3.2 Reward Function Effectiveness

At the same time, to prove that the reward function we designed based on the local observation form is effective, we test the evaluation metrics in Section 6.2 on the trained model to verify that our reward function can satisfy equation (18). As shown in Fig. 13, with the increase of cumulative reward, the message redundancy index MR decreases at the end of broadcasting, the propagation delay index PL decreases, and the redundancy delay score RL increases. This experiment proved that our designed reward function enables the UAV to learn a good gossip propagation strategy, resulting in an overall optimization effect.

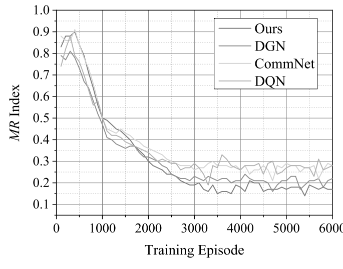
6.4 Performance Comparison

In this section, we will verify the performance of the proposed scheme in terms of message broadcasting and compare it with four broadcasting approaches, namely BFBA, RGBA, CFPBA, and SFNBA. Specifically, we will measure three aspects: the required delay for propagating, the number of redundant messages, and power consumption.

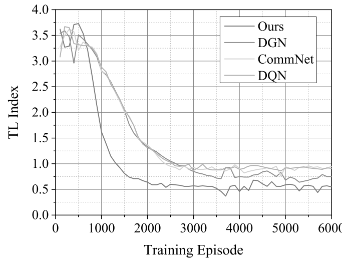
6.4.1 Propagation Delay Comparison

Fig. 14 shows the number of rounds required for the five broadcasting methods to spread the message throughout the network (note: for BFBA and RGBA, we only count when the message is broadcast successfully). According to the theoretical analysis in [10], [12], compared to other methods, BFBA has an optimal convergence speed with only five rounds of propagation because each UAV sends the message packet to all its neighbors instead of one neighbor when forwarding. Our experimental results also verify the above conclusions. RGBA and CFPBA are significantly higher than BFBA in terms of propagation delay because they are still random in forwarding message packets. Because CFPBA introduces a forwarding strategy with decreasing probability of redundant messages, it avoids sending part of the redundant messages and can slightly reduce the propagation rounds. SFNBA introduces some fixed strategies in selecting neighbors, which further accelerates the convergence speed and propagates the message packets. SFNBA further accelerates the convergence speed, and the propagation rounds are only 11. Compared with other schemes, the model is trained to broadcast so that the probability and neighbors are not completely random when forwarding message packets. The forwarding strategy learned by the model makes the UAV forward the message packets with a higher probability to neighboring UAVs that have never received a message packet. Hence, our scheme's message packet forwarding is more efficient than randomly selecting a neighbor. Experiments show that when the number of UAVs is 125, the required number of rounds for our scheme is 20, which is better than 26 for CFPBA and 25 for SFNBA.

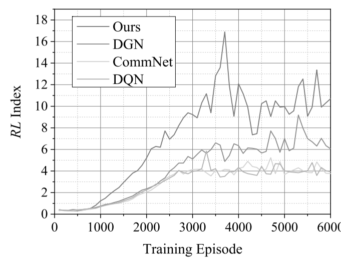
We also record the specific dealy required for the five broadcasting methods at the corresponding rounds. Fig. 15 shows the required delay at the different number of UAVs, and on average, our scheme requires a delay saving of 36.1%



(a) Message Redundancy Index



(b) Propagation Delay Index



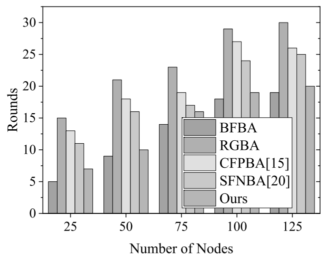


Fig. 14. Comparison of the number of rounds for broadcasting.

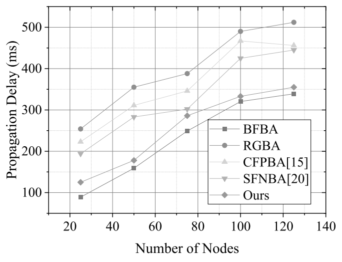


Fig. 15. Comparison of the delay required for broadcasting.

compared to RGBA, 29.2% compared to CFPBA, and 22.6% compared to SFNBA.

6.4.2 Redundant Packets Comparison

Fig. 16 shows the number of redundant packets for different numbers of UAVs (note: for BFBA and RGBA, we only count when the message broadcast is successful). From the figure, we can see message implosion due to the flood approach used by BFBA in propagating the message packets. Since we fixed the spatial size of the UAV distribution, as the number of UAVs increases, the network density then increases, leading to a rise in the number of neighbors per UAV, thus making the message implosion appear more severe.

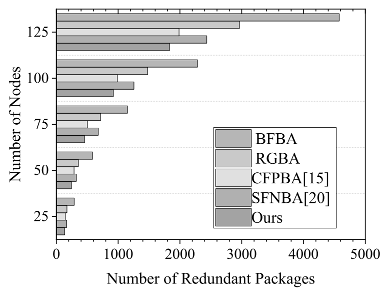


Fig. 16. Comparison of the redundant packets for broadcasting.

Since RGBA randomly selects three neighbors for message packet forwarding, the increase in the number of UAVs does not make the number of redundant packets rise sharply. Experiments show that when the number of UAVs is 125, CFPBA significantly reduces the forwarding probability for redundant message packets, so the number of redundant packets can be reduced considerably by 32.9% compared to RGBA. Since only neighbor selection is considered, when in the face of redundant packets, SFNBA still forwards redundant packets with probability 1, so the number of redundant packets is only reduced by 17.8% compared to RGBA. Our trained model first sends packets to the desired UAVs, which improves the propagation efficiency and minimizes the forwarding probability or even does not send packets when multiple duplicate packets are received. So redundant messages of our scheme are reduced by 38.2% compared to RGBA, 6.9% to CFPBA, and 20.7% to SFNBA.

6.4.3 Power Consumption Comparison

The reception and processing of redundant packets exacerbate the power consumption, and we compare only the power consumption of flood, gossip, and our scheme,

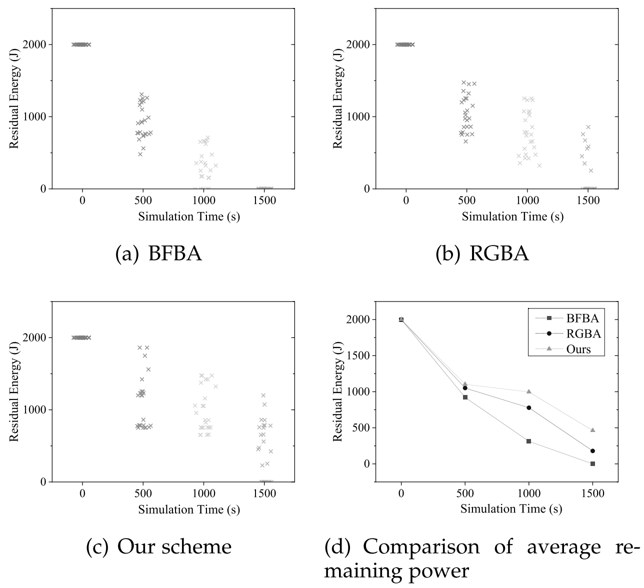


Fig. 17. Comparison of remaining power for 25 UAVs.

where we measure the energy consumption of each UAV during message broadcasting. For example, there are 25 UAVs in the network, and each UAV battery can supply 2000 joules of energy. Fig. 17(a)(b)(c) shows the different schemes' power consumption at different simulation times. It is clear that due to a large number of redundant packets in BFBA, all UAVs are in a state of power depletion at 1500 seconds in a continuous message broadcast scenario. However, compared to BFBA, RGBA still has eight UAVs surviving, and our scheme still has 17 UAVs surviving. Besides, we averaged the remaining power of the UAVs at different times to make a comparison of the three schemes. Fig.17(d) indicates that our scheme can save more power.

6.4.4 Application Case: P2P aerial photography service for disaster relief

To further validate the superiority of our proposed scheme, we apply the proposed protocol to gossip-based P2P live video streaming, which is a distributed video-on-demand system such as QvodPlayer [41]. Such systems are designed to provide smooth video services to users with limited bandwidth. In a disaster such as an earthquake, clusters of UAVs are sent to areas that are difficult to reach by humans to conduct an aerial search. To ensure that rescue is carried out in time, aerial video needs to be transmitted back to various units on the ground for viewing without delay [42].

We set ten UAVs in the sky for aerial photography and five rescue units on the ground for accessing aerial photography data. The UAVs adopts the communication mode of 802.11b with a bandwidth of 11 Mbps. It is worth noting that multi-access concurrency and video streaming lead to spikes and cause bandwidth usage fluctuations in Fig. 18. In Fig. 18 and Fig. 19, when the five units request 720P vid

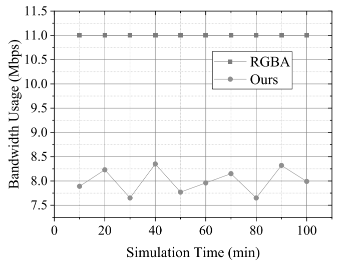


Fig. 18. Comparison of bandwidth usage.

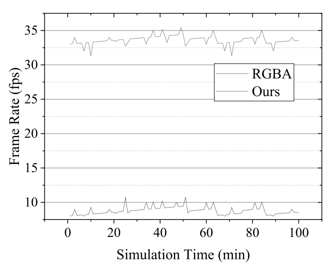


Fig. 19. Comparison of video frame rates.

frame rate is greater than 30fps. The frame rate of the returned video should be at least 30fps [43] so as to provide reliable aerial video for the ground units.

7 Conclusion

This paper proposes an intelligent adaptive gossip-based broadcast protocol using multi-agent reinforcement learning for UAV-MEC. First, we design a data structure called Bitgraph while laying the foundation for the subsequent design of reward functions. Second, we propose a reward function and design the BDGN algorithm to make simultaneous decisions on forwarding probability and neighbors. The heuristic ideas behind the scheme are: "people tend to tell people who are not yet informed when spreading the news" and "People lose the desire to spread the news that is widely circulated". We intend to guide the agents in learning a good forwarding strategy in a dynamic environment. Extensive experimental results show that our scheme can achieve efficient broadcasting.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62125205, the Key Research and Development Program of Shaanxi under Grant 2021ZDLGY05-04, the Fundamental Research Funds for the Central Universities under Grant XJSJ23188, and the Fundamental Research Funds for the Central Universities No. YJSJ23007.

References

[1] Research and Markets, "Edge computing market size, share trends analysis report," Website, 2022, https://www.grandviewresearch.com/industry-analysis/edge-computing-market Accessed Novermber 22, 2022.

[2] Y. Du, K. Wang, K. Yang, and G. Zhang, "Energy-efficient resource allocation in uav based mec system for iot devices," in *2018 IEEE Global Communications Conference (GLOBECOM'18)*. IEEE, 2018, pp. 1–6.

[3] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, "Uav-assisted mec networks with aerial and ground cooperation," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7712–7727, 2021.

[4] M. Samir, C. Assi, S. Sharafeddine, and A. Ghrayeb, "Online altitude control and scheduling policy for minimizing aoi in uav-assisted iot wireless networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2493–2505, 2020.

[5] L. Zhang, A. Celik, S. Dang, and B. Shihada, "Energy-efficient trajectory optimization for uav-assisted iot networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4323–4337, 2021.

[6] M. Dai, Y. Wu, L. Qian, Z. Su, B. Lin, and N. Chen, "Uav-assisted multi-access computation offloading via hybrid noma and fdma in marine networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 113–127, 2022.

[7] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in uav-assisted multi-access edge computing systems under resource uncertainty," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 175–190, 2021.

[8] Z. Ning, Y. Yang, X. Wang, L. Guo, X. Gao, S. Guo, and G. Wang, "Dynamic computation offloading and server deployment for uav-enabled multi-access edge computing," *IEEE Transactions on Mobile Computing*, 2021.

[9] H. Song, L. Liu, B. Shang, S. Pudlewski, and E. S. Bentley, "Enhanced flooding-based routing protocol for swarm uav networks: random network coding meets clustering," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[10] R. Yang, "Analysing the efficiency and robustness of gossip in different propagation processes with simulations," in *2019 4th International Seminar on Computer Technology, Mechanical and Electrical Engineering (ISCME'19)*, vol. 1486, no. 3. IOP Publishing, 2020, p. 032001.

[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th international conference on Supercomputing (ICS'02)*, 2002, pp. 84–95.

[12] E. Leme, N. Ivaki, N. Laranjeiro, and R. Moraes, "Analyzing gossip protocols for reliable manet applications," in *2017 IEEE International Conference on Edge Computing (EDGE'17)*. IEEE, 2017, pp. 98–105.

[13] P. Felber, A.-M. Kermarrec, L. Leonini, E. Riviere, and S. Voulgaris, "Pulp: An adaptive gossip-based dissemination protocol for multi-source message streams," *Peer-to-Peer networking and Applications*, vol. 5, no. 1, pp. 74–91, 2012.

[14] P. Kyasanur, R. R. Choudhury, and I. Gupta, "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks," in *2006 IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 2006, pp. 91–100.

[15] W. Cheng and L. Zhang, "Research on gossip algorithms with binary exponential backoff," *Journal of Electronics Information Technology*, vol. 43, no. 12, pp. 3486–3495, 2021.

[16] D. Tian, J. Zhou, Y. Wang, G. Zhang, and H. Xia, "An adaptive vehicular epidemic routing method based on attractor selection model," *Ad hoc networks*, vol. 36, pp. 465–481, 2016.

[17] M. Matos, V. Schiavoni, P. Felber, R. Oliveira, and E. Riviere, "Lightweight, efficient, robust epidemic dissemination," *Journal of Parallel and Distributed Computing*, vol. 73, no. 7, pp. 987–999, 2013.

[18] O. Ozkasap, E. Cem, S. E. Cebeci, and T. Koc, "Flat and hierarchical epidemics in p2p systems: Energy cost models and analysis," *Future Generation Computer Systems*, vol. 36, pp. 257–266, 2014.

[19] C. Esposito, A. Castiglione, F. Palmieri, and M. Ficco, "Improving the gossiping effectiveness with distributed strategic learning," *Future Generation Computer Systems*, vol. 71, pp. 221–233, 2017.

[20] L. Altoaimy, H. Kurdi, A. Alromih, A. Alomari, E. Alrogi, and S. H. Ahmed, "Enhanced distance-based gossip protocols for wireless sensor networks," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC'19)*. IEEE, 2019, pp. 1–4.

[21] A. G. Barto, R. S. Sutton, and C. Watkins, *Learning and sequential decision making*. University of Massachusetts Amherst, MA, 1989.

[22] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *International conference on machine learning (PMLR'18)*. PMLR, 2018, pp. 4344–4353.

[23] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'18)*, vol. 32, no. 1, 2018.

[24] O. Marom and B. Rosman, "Belief reward shaping in reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'18)*, vol. 32, no. 1, 2018.

[26] D. Cason, N. Milosevic, Z. Milosevic, and F. Pedone, "Gossip consensus," in *Proceedings of the 22nd International Middleware Conference (Middleware'21)*, 2021, pp. 198–209.

[27] Z. Xu, K. Ye, X. Dong, H. Han, Z. Yan, X. Chen, D. Liao, and H. Wang, "Dc-gossip: An enhanced broadcast protocol in hyperledger fabric based on density clustering," in *Wireless Algorithms, Systems, and Applications: 17th International Conference, WASA 2022, Dalian, China, November 24–26, 2022, Proceedings, Part III*. Springer, 2022, pp. 3–19.

[28] G. Saldamli, C. Upadhyay, D. Jadhav, R. Shrishrimal, B. Patil, and L. Tawalbeh, "Improved gossip protocol for blockchain applications," *Cluster Computing*, vol. 25, no. 3, pp. 1915–1926, 2022.

[29] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing (PODC'87)*, 1987, pp. 1–12.

[30] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, vol. 34, 2016.

[31] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *The International Conference on Learning Representations (ICLR'18)*, 2018.

[32] Z. Ye, K. Wang, Y. Chen, X. Jiang, and G. Song, "Multi-uav navigation for partially observable communication coverage by graph reinforcement learning," *IEEE Transactions on Mobile Computing*, early access, 31 January, 2022, doi: 10.1109/TMC.2022.3146881.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems (NIPS'17)*, vol. 30, no. 1, 2017.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *The International Conference on Learning Representations (ICLR'18)*, 2018.

[36] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence (AAAI'16)*, vol. 30, no. 1, 2016.

[37] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li, "Colight: Learning network-level cooperation for traffic signal control," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1913–1922.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[39] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.

[40] S. Sukhbaatar, R. Fergus *et al.*, "Learning multiagent communication with backpropagation," *Advances in neural information processing systems*, vol. 29, 2016.

[41] C. Guo, "Analyse the penalty limits of the provider of p2p sharing technology," in *2021 4th International Conference on Humanities Education and Social Sciences (ICHESS'2021)*. Atlantis Press, 2021, pp. 1187–1193.

[42] M. Półka, S. Ptak, and Ł. Kuziora, "The use of uav's for search and rescue operations," *Procedia engineering*, vol. 192, pp. 748–752, 2017.

[43] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, 2008.

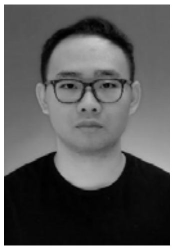
Zhe Ren received the B.S. degree in computer science and technology from Xidian University in 2014. He is currently working toward the Ph. D. degree in security of cyberspace in Xidian University. His research interests include wireless protocol design, mobile network security and multi-agent reinforcement learning.

Xinghua Li (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Xidian University in 2004 and 2007, respectively. He is currently a professor in the School of Cyber Engineering, Xidian University, China. His research interests include wireless networks security, and protocol formal methodology.

Yinbin Miao received the B.E. degree from the Department of Telecommunication Engineering, Jilin University, Changchun, China, in 2011, and the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He is currently an Associate Professor with the Department of Cyber Engineering, Xidian University. His research interests include information security and applied cryptography.

Zhuowen Li received the B.S. degree in information security from Xidian University in 2021. He is currently working toward the M.S. degree in Cyber Engineering in Xidian University. His research interests include blockchain technology, unmanned aerial vehicle networks and deep reinforcement learning.

Zihao Wang received the B.S. degree in internet of things engineering from University of Electronic Science and Technology of China in 2021. He is currently working toward the M.S. degree in Security of Cyberspace in Xidian University. His research interests include blockchain, reinforcement learning, and payment channel networks.



Mengyao Zhu received the B.S. and M.E. degrees in cyberspace security from Xidian University in 2019 and 2022, respectively. He is currently a research assistant with the School of Cyber Engineering at Xidian University. His main research interests include artificial intelligence security and information security.

Ximeng Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi'an, China. He was a Research Fellow at the School of Information System, Singapore Management University, Singapore. He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University. His research interests include cloud security, applied cryptography, and big data security.

Robert H. Deng (Fellow, IEEE) is currently the AXA Chair Professor of cybersecurity, the Director of the Secure Mobile Centre, and the Deputy Dean of the Faculty and Research, School of Computing and Information Systems, Singapore Management University. His research interests include data security and privacy, network security, and applied cryptography. He is a fellow of the Academy of Engineering Singapore. He received the Outstanding University Researcher Award from the National University of Singapore, the Lee Kuan Yew Fellowship for Research Excellence from SMU, and the Asia–Pacific Information Security Leadership Achievements Community Service Star from the International Information Systems Security Certification Consortium. He serves/served on the Editorial Board of the ACM Transactions on Privacy and Security, IEEE Security and Privacy, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, and Journal of Computer Science and Technology, and the Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security.