EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Intelligent deep reinforcement learning-based scheduling in relay-based HetNets

Chao Chen[1], Zhengyang Wu[1], Xiaohan Yu[1], Bo Ma[1] and Chuanhuang Li[1*]

*Correspondence: chuanhuang_li@zjgsu.edu.cn

[1] School of Information and Electronic Engineering (Sussex Artificial Intelligence Institute), Zhejiang Gongshang University, Hangzhou 310018, China

Abstract

We consider a fundamental file dissemination problem in a two-hop relay-based heterogeneous network consisting of a macro base station, a half-duplex relay station, and multiple users. To minimize the dissemination delay, rateless code is employed at the base station. Our goal is to find an efficient channel-aware scheduling policy at the half-duplex relay station, i.e., either fetch a packet from the base station or broadcast a packet to the users at each time slot, such that the file dissemination delay is minimized. We formulate the scheduling problem as a Markov decision process and propose an intelligent deep reinforcement learning-based scheduling algorithm. We also extend the proposed algorithm to adapt to dynamic network conditions. Simulation results demonstrate that the proposed algorithm performs very close to a lower bound on the dissemination delay and significantly outperforms baseline schemes.

Keywords: Channel-aware scheduling, Heterogeneous network, Rateless code, Deep reinforcement learning

1 Introduction

With the increasing demand for high-speed data transmission, it is increasingly important to improve network capacity. One promising solution that has garnered significant attention recently is the heterogeneous networks (HetNets). By deploying different types of elements, such as picocells, femtocells, and relay stations, within the macrocells coverage, HetNets can increase the potential for spatial resource reuse and enhance the quality of service (QoS) of wireless users [1]. Meanwhile, due to the explosive growth of multimedia traffic, broadcasting/multicasting has recently received much interest as an efficient means for dissemination of multimedia information to wireless users, e.g., 5G New Radio Multimedia Broadcast/Multicast Service (NR-MBMS) [2, 3].

In this paper, we consider a fundamental problem of minimum-delay file dissemination in two-hop relay-based HetNets. As shown in Fig. 1, the system is composed of a macro base station (BS), a half-duplex relay station (RS), and multiple wireless users associated with the RS. Such a two-hop relaying structure is a key building block from which to construct versatile networks in 5G/6G [4, 5]. The BS intends to disseminate

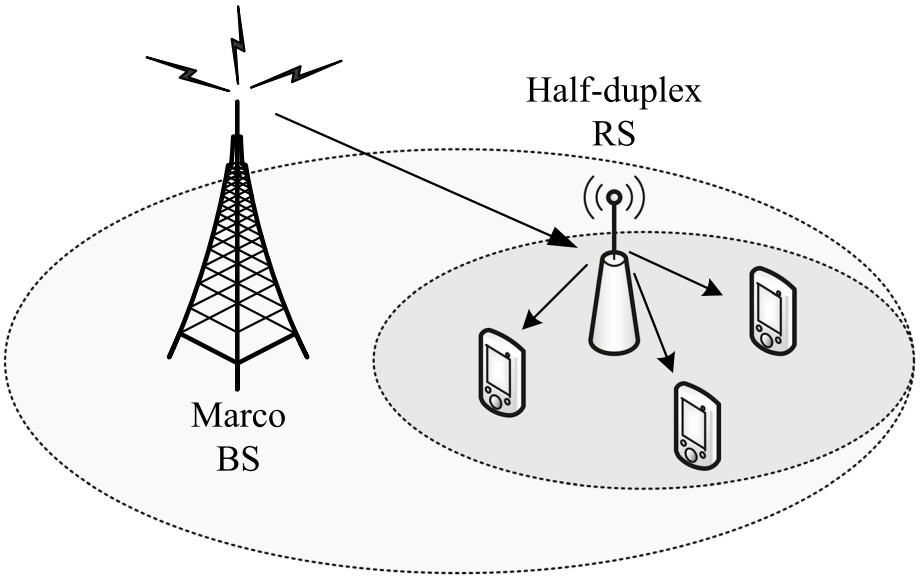

Fig. 1 Relay-based HetNet

a block of packets to all users over time-varying channels. To minimize the dissemination delay, rateless code [6] is employed at the BS, such that coded packets are disseminated to the users, and the original packets can be decoded by a user once a sufficient number of coded packets are received.

At each time slot, the half-duplex RS can either broadcast a coded packet to the users or fetch a coded packet from the BS. Due to the dynamic nature of the wireless environment, at each time slot, only a subset of users are able to receive a transmitted packet successfully. The goal is to minimize the file dissemination delay by making optimal channel-aware scheduling decisions at the RS, i.e., either fetching or broadcasting a packet at each time slot.

Due to the time-varying nature of wireless channels, it is desirable for the RS to broadcast a packet if the overall channel states between the RS and the users are favorable, e.g., when most of the users are able to receive the broadcast packet at the current time slot successfully. Note that, the users' channel conditions are typically asymmetric in practice, such that the dissemination delay is usually bottlenecked by the users with poor channel conditions. Hence, the RS should also pay more attention to the users with poor channel conditions. Moreover, making a decent balance between fetching and broadcasting packets at the RS is important, such that the RS buffer can be replenished with new packets in time during the transmission, and also the users can receive packets with high rates. However, it is challenging to make optimal scheduling decisions at the RS, even for small-sized systems (i.e., the number of users is small).

Our contributions are summarized below. We study a channel-aware scheduling problem for efficient file dissemination in two-hop relay-based HetNets. We model the scheduling problem at the half-duplex RS as a Markov decision process (MDP) and propose a deep reinforcement learning (DRL)-based intelligent scheduling

algorithm. The proposed scheduling algorithm does not rely on the knowledge of statistic channel parameters, and can be used in general scenarios where the BS-to-RS channel is not ideal. We also extend the proposed algorithm to adapt to the network dynamics. Through simulations, we show that the proposed algorithm performs close to a lower bound on dissemination delay and significantly outperforms baseline schemes such as the uncoded Automatic Repeat-reQuest (ARQ).

The organization of this paper is summarized below. The related work is presented in Sect. 2. Section 3 describes the system model. The MDP formulation of our problem and the proposed scheduling algorithm are presented in Sect. 4. Simulation results are provided in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Related work

There have been numerous studies on multicast/broadcast in single-hop networks. In [7], the authors provide precise bounds on the maximum throughput after analyzing the throughput performance of rateless code in single-hop broadcast networks. The study in [7] is focused on symmetric channel distributions; however, in our study, we consider a generic asymmetric channel distribution. Sim et al. [8] and Low et al. [9] study opportunistic coded multicast scheduling policies over a single-hop network, where the key trade-off is between multi-user diversity and multicast gain. Khamfroush et al. [10] proposes a scheduling policy for the transmission of coded packets in cooperative networks. However, the authors in [10] employ a dynamic programming-based approach, that is computationally intractable when the network size increases. Coded multicast scheduling over relay-based HetNets is investigated in [11], where the authors employ fluid relaxation to formulate the scheduling problem at the half-duplex RS. A threshold-based policy termed WBP-TRACK is proposed, and a lower bound on dissemination delay is derived. However, they only consider the special case where the BS-to-RS channel is perfect, which is different from ours.

Recently, reinforcement learning (RL) has seen wide spread applications for wireless communication networks, particularly in decision-making problems [12, 13]. Iqbal et al. [14] studies the adaptive resource allocation scheme based on Q-learning in HetNets. The cell handover problem in HetNets is studied in [15], where a context-aware scheduling algorithm based on RL is proposed. The authors in [16] propose a sleep scheduling algorithm based on RL for compressive data gathering, which aims to enhance the energy efficiency and prolong the lifespan of wireless sensor networks. The authors in [17] investigate a downlink power allocation problem in HetNets and propose power allocation algorithms based on Q-learning. The spectrum sharing issue in cognitive radio networks is studied in [18] using RL, where the authors propose an RL-based channel sensing and selection algorithm, which does not rely on the knowledge of system parameters. The above works are all based on the framework of RL. However, when the state and/or action space grows large, the effectiveness of RL decreases. In that case, DRL becomes a more appropriate choice for decision-making problems. Tang et al. [19] proposes a DRL-based algorithm for real-time wireless resource allocation in high mobility HetNets. The authors in [20] study efficient power allocation in HetNets and propose a DRL-based power allocation algorithm. The work [21] considers end-to-end network slicing enabling HetNets, and proposes a switching algorithm using DRL. Ma

and Wong [22] and Lie t al. [23] investigate dynamic content caching in HetNets and propose efficient DRL-based scheduling algorithms for updating cache content. Cong and Lang [24] studies the dynamic multichannel access problem in multi-user wireless networks, and proposes a DRL-based algorithm to enhance the spectrum utilization in dynamic spectrum access. In [25], the authors consider a joint relay selection and power allocation problem in two-hop cooperative relay networks, and propose a hierarchical reinforcement learning-based policy to minimize the outage probability. However, they simply assume a fixed operation pattern at half-duplex relays, i.e., one time slot for fetching a packet and the next time slot for broadcasting a packet, which is different from our paper.

In summary, the existing scheduling schemes used in relay-based HetNets either have high computational complexity or only apply to systems with special structures. By contrast, in this paper, we consider a general system model and propose a DRL-based scheduling scheme that surpasses the performance of existing schemes.

3 System model

As shown in Fig. 2, we consider a relay-based HetNet in which a macro BS aims to disseminate a common data file to a group of N users via a half-duplex RS. The set of user indices is denoted as $I := \{1, \ldots, N\}$. The data file contains K original (raw) packets, each of which can be represented as a vector of length v over a finite coding field $\mathbb{F}_d$ where d is the size of the coding field. Hence, each packet has a length of $v\lceil \log_2 d \rceil$ bits. To minimize the file dissemination delay, rateless code is employed at the BS. That is, an infinite number of coded packets are generated and backlogged at the BS buffer initially. The buffers of the RS and users are initially empty. We consider a time-slotted system, where the half-duplex RS can either fetch a coded packet from the BS or broadcast a previously received coded packet to the users at each time slot. Once a packet is broadcast by the RS, it will be removed from the RS buffer. For the sake of simplicity, we assume that once a user receives any K coded packets, it can recover (decode) the original packets, and the coding overhead is negligible [26].

We assume that the RS broadcasts packets utilizing a single frequency-time resource block (RB) and a predetermined modulation and coding scheme (MCS). Let the random process $C_i(t) \in \mathbb{Z}_0$ denote the number of bits that can be successfully transmitted from the RS-to-user i on the transmission RB at time slot t. We assume

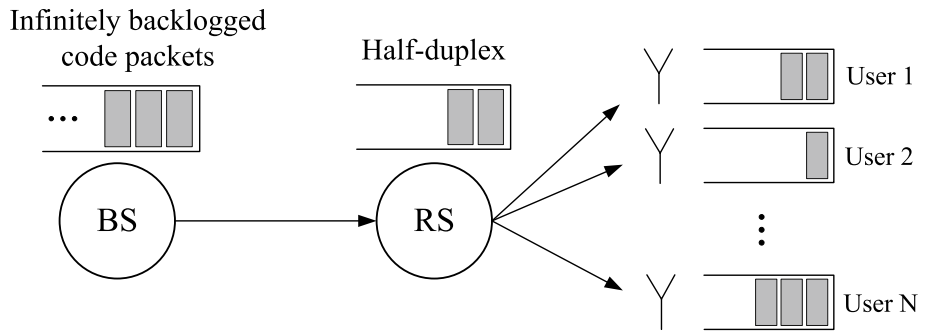

Fig. 2 System model

that users' channels undergo fast fading, i.e., $C_i(\cdot)$ varies independently over time slots for all $i \in I$. If the RS chooses to broadcast a packet at time slot t, two possibilities can occur for the user i: it either successfully receives the packet, i.e., $C_i(t)$ is greater than the packet length $\nu\lceil \log_2 d \rceil$, or it fails to receive the packet. Thus, we define users' channel states as binary random processes

$$X_i(t) := \mathbf{1}(C_i(t) \geq \nu\lceil \log_2 d \rceil), \forall i \in I, \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function. Thus, the channels are inherent 'ON-OFF' channels. In the sequel, we say that the RS-to-user i channel is in ON (resp. OFF) state if user i can successfully receive a packet at time slot t, i.e., $X_i(t) = 1$ (resp. $X_i(t) = 0$). Similarly, we let $C_0(\cdot)$ denote the number of bits that can be successfully transmitted from the BS to the RS at time slot t, and define the BS-to-RS channel's state as a binary random process

$$X_0(t) := \mathbf{1}(C_0(t) \geq \nu\lceil \log_2 d \rceil). \tag{2}$$

Let $X(t) := (X_0(t), X_1(t), \ldots, X_N(t))$ denote the aggregate channel state information (CSI) at time slot t. We assume that CSIT such that $X(t)$ is known at the RS at the start of each time slot t. The probability that the RS-to-user i channel is in ON state at time slot t is denoted by $p_i := \mathbb{P}(X_i(t) = 1)$, and the probability that the BS-to-RS channel is in ON state at time slot t is denoted by $p_0 := \mathbb{P}(X_0(t) = 1)$. Let $p := (p_0, p_1, \ldots, p_N)$ denote the aggregate channel parameters of the system.

Let $B_i(t)$ (resp. $B_0(t)$) denote the number of coded packets backlogged in the buffer of user i (resp. the RS) at time slot t. Define $B(t) := (B_0(t), B_1(t), \ldots, B_N(t))$ as the aggregate buffer state information (BSI) of the system at time slot t. $B(t)$ can be easily tracked over time by the RS based on the CSI and scheduling decisions made in previous time slots.

The goal is to minimize the dissemination delay, which is defined as the number of time slots it takes for all users to receive at least K coded packets, through making optimal scheduling decisions (fetching a packet from the BS or broadcasting a packet to the users) at each time slot. Table 1 provides a list of the major notations used in this work.

Table 1 Summary of the notation

Notation	Description
N	Number of users
I	$:= \{1, \ldots, N\}$, set of user indices
K	Number of packets in a data block
d	Coding field size
$X_0(t)$	BS-to-RS channel's state at time slot t
$X_i(t)$	RS-to-user i channel's state at time slot t
$X(t)$	$:= (X_0(t), X_1(t), \ldots, X_N(t))$, aggregate CSI of the system at time slot t
$B_0(t)$	Number of coded packets backlogged at the buffer of the RS at time slot t
$B_i(t)$	Number of coded packets backlogged at the buffer of user i at time slot t
$B(t)$	$:= (B_0(t), B_1(t), \ldots, B_N(t))$, aggregate BSI of the system at time slot t
p_0	Probability that the BS-to-RS channel is in ON state at each time slot
p_i	Probability that the RS-to-user i channel is in ON state at each time slot
p	$:= (p_0, p_1, \ldots, p_N)$, aggregate channel parameters

4 Methods

In this section, we first model our problem as a Markov decision process (MDP), and then propose an intelligent DRL-based scheduling policy.

Our problem is a complex decision-making problem in stochastic networks, where the aggregate channel parameters p may not be known to the RS. Generally, when the system model is unknown, there are two main approaches to solving decision-making problems: model-based and model-free methods [27]. In the model-based method, the decision maker first estimates the system model according to the observations and then applies a dynamic programming or heuristic policy based on the estimated system model. By contrast, in the model-free method, the decision maker learns a policy directly through interactions with the system without estimating the system model. The model-based method may not be applicable due to incorrect estimations of the system model caused by the limited observation ability of the decision maker. Moreover, even if the system model can be accurately estimated, solving a decision-making problem with a large model size is typically computationally intractable. Thus, in this paper, we will adopt the model-free method to solve our problem.

4.1 MDP formulation

The MDP formulation of our problem can be described as follows:

(i) System State

The system state at time slot t, denoted by $s(t)$, is defined as

$$\begin{aligned} s(t) &= \{B(t), X(t)\} \\ &= \{B_0(t), B_1(t), ..., B_N(t), X_0(t), X_1(t), ..., X_N(t)\}. \end{aligned} \tag{3}$$

That is, the system state includes both the aggregate CSI and BSI. Let $\mathbf{S}$ denote the state space. The RS selects an appropriate action at the beginning of each time slot according to the current system state $s \in \mathbf{S}$.

(ii) Action

Since RS operates in the half-duplex mode, the RS can either broadcast or fetch a packet at each time slot. Therefore, the action a is defined as a binary number such that $a = 0$ (resp. $a = 1$) denotes that the RS chooses to broadcast (resp. fetch) a packet. The action space $\mathbf{A}$ is given by $\{0, 1\}$. Note that, it is optimal to fetch a packet when the RS buffer is empty, i.e., $B_0(t) = 0$.

(iii) Reward

The *bottleneckuser* is defined as the one with the fewest buffered packets. Note that there may exist multiple bottleneck users at each time slot. For given $s \in \mathbf{S}$ and $a \in \mathbf{A}$, the reward is defined as

$$r(s,a) = \begin{cases} 1000, & \text{if the block transmission is completed,} \\ 1, & \text{if one of bottleneck users receives a packet,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Intuitively, the reward is set to a high value (1000) if all users receive K coded packets and hence successfully decode the original data block, i.e., $\min_{i \in I}\{B_i(t)\} \geq K$; if one of the bottleneck users receives a packet, the reward is set to 1. This is because,

in practical systems, the users are typically heterogeneous and the dissemination delay is usually dominated by the performance of the bottleneck user. In the sequel, we use $r(t)$ to denote the reward received at time slot t.

(iv) Transition Probability

$p_{ss'}(a)$ denotes the transition probability from the present system state s to another system state s' under action a. Specifically, for given $s = \{B, X\}$ and $s' = \{B', X'\}$, if the following conditions are satisfied

$$\begin{aligned} B'_0 &= \max\{B_0 - a + (1+a)X_0, 0\}, \\ B'_i &= B_i + aX_i(B_0 > 0), \forall i \in I, \end{aligned} \tag{5}$$

the transition probability is given by

$$p_{ss'}(a) = \prod_{i=0}^{N} p_i^{B'_i}(1-p_i)^{1-B'_i}. \tag{6}$$

Otherwise, we have that $p_{ss'}(a) = 0$.

4.2 Deep reinforcement learning

DRL is an efficient model-free method for decision-making problems that can learn optimal policies through trial-and-error. It allows learning the optimal policies directly from observations without estimating the system model, making it suitable for large and complex systems.

The goal of DRL is to adopt a policy to maximize long-term return

$$G_t = \sum_{\tau=0}^{\infty} \gamma^{\tau} r(t+\tau+1), \tag{7}$$

where $\gamma \in [0, 1]$ is the discount factor for future rewards. Moreover, DRL defines the state–action value function

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | s, a], \tag{8}$$

under policy π, and iteratively solves the Bellman Equation

$$q_{\pi}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}(a) \left[\sum_{a' \in \mathbf{A}} \pi(a'|s') q_{\pi}(s', a') \right]. \tag{9}$$

The optimal policy,π^*, can be derived as

$$\pi^*(s) = \arg\max_{a \in \mathbf{A}} q_{\pi^*}(s, a). \tag{10}$$

Deep Q-network (DQN) is the most representative method within the field of DRL [28]. It involves using a neural network (referred to as the Q-value network) with weight parameters w to approximate the state–action value function. DQN is an empirical value iteration approach that seeks to find the Q-value of each state–action pair (s, a).

DQN takes a state–action pair as the input and outputs the corresponding Q-value $Q_\pi(s, a; w)$. Through the analysis of the historical states, actions, and rewards, DQN iteratively updates the weights of the neural network w to minimize the loss function

$$\mathbb{L}(w) = [U - Q_\pi(s, a; w)]^2, \tag{11}$$

where $U = r(s, a) + \gamma \max_a Q_\pi(s', a; w)$ is the target output of DQN [29].

Since the Q-value update of DQN includes a maximization operation, it can lead to an overestimation of the Q-value. Dueling Double DQN (Dueling-DDQN) addresses this issue by combining the approaches of Double DQN and Dueling DQN as follows. Firstly, Double DQN adopts different update formulas for action selection and evaluation, which helps reduce the overestimation error. It involves the use of a double network structure [30], consisting of an evaluation network $Q_{\text{eval}}(s, a; w)$ for action selection, i.e., $a_{\max} = \arg\max_a Q_{\text{eval}}(s', a; w)$, and a target network $Q_{\text{targ}}(s, a; w')$ for action evaluation, such that

$$U = r(s, a) + \gamma Q_{\text{targ}}(s', \arg\max_{a \in \mathbf{A}} Q_{\text{eval}}(s', a; w); w'). \tag{12}$$

Secondly, the Dueling DQN separates the Q-value into two components: a state value function $V(s)$ that only depends on the state, and an advantage function $A(s, a)$ that depends on both the state and the action [31], i.e.,

$$Q(s, a; w, \alpha, \beta) = V(s; w, \alpha) + A(s, a; w, \beta), \tag{13}$$

where α and β are the unique parameters associated with $V(s)$ and $A(s, a)$, respectively.

4.3 Intelligent DRL-based scheduling policy

In this subsection, we adopt Dueling-DDQN to learn the optimal scheduling policy at the half-duplex RS to minimize the file dissemination delay. Figure 3 shows the structure of the proposed Intelligent DRL-based Scheduling (IDS) algorithm. It consists of four functional modules at the RS: a DRL agent, a transmitter/receiver module, an experience replay memory, and an action selection module.

At the start of each time slot t, the DRL agent observes the current state $s(t) = \{B(t), X(t)\}$ and receives a reward $r(t-1)$ based on the last state $s(t-1)$. Then, the DRL agent generates an experience $e(t) = \langle s(t-1), a(t-1), r(t-1), s(t)\rangle$ based on the state, action and reward of the previous time slot. Once an experience is obtained, it is saved in the experience replay memory, which uses a first-in-first-out replacement policy. That is, when the experience replay memory is full, the oldest experience will be removed. Meanwhile, the DRL agent inputs the current state $s(t)$ to the action selection module, which uses the ϵ-greedy algorithm to select an action,$a(t)$, for the current time slot: with probability ϵ, the action is chosen uniformly from the action space; with probability $1 - \epsilon$, the action with the highest Q-value, as determined by the evaluation network, is selected. The selected action $a(t)$ is then executed at the transmitter/receiver module.

Meanwhile, at each time slot, the DRL agent randomly chooses a mini-batch of experience samples, denoted as Z, from the experience replay memory and inputs them into

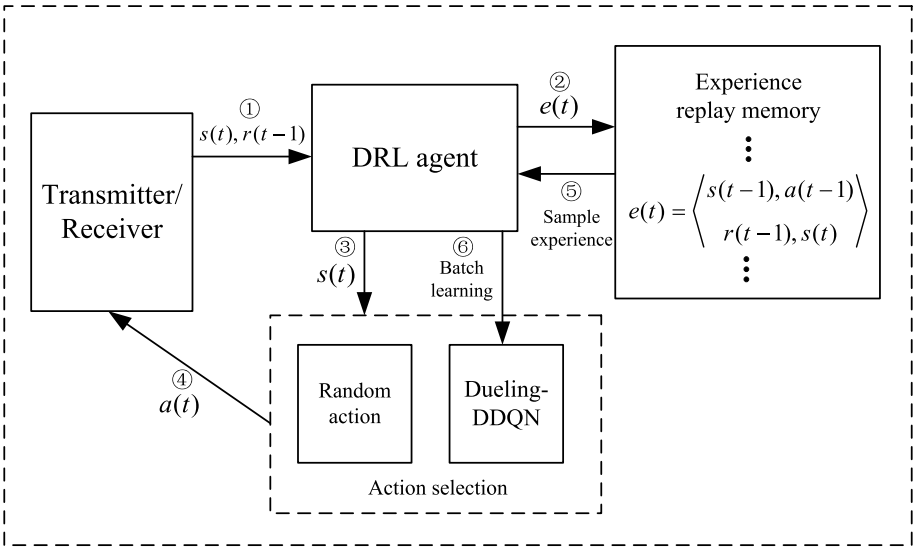

Fig. 3 The structure of IDS algorithm

the Dueling-DDQN in the action selection module for batch training, i.e., updating the weights of the evaluation network. This allows the agent to efficiently learn from multiple experiences at once and avoid the inefficiency of training on a single experience sample. Since the experience replay memory is empty initially, batch training will not be carried out in the first T time slots. The weights of the target network, w^- are updated (set as to the weights of the evaluation network w) every M time slot. Once the evaluation network has converged, set $Q_{\text{targ}} = Q_{\text{eval}}$, and Q_{targ} is the optimal Q-value network found by IDS. The RS will then be scheduled according to Q_{targ}. The details of the proposed algorithm are shown in Algorithm 1.

Algorithm 1 Intelligent DRL-based Scheduling (IDS)

Establish the evaluation network and the target network.
Initialize the weights of evaluation network w randomly, and set the weights of target network w^- as w. Initialize T, M, Z.
In each time slot $t \leq T$, the DRL agent randomly chooses an action to execute and saves the experience $e(t)$ in the experience replay memory.
for $t = T+1, T+2, \ldots$ **do**
    The DRL agent observes the current state $s(t)$ and the reward of last time slot $r(t-1)$.
    The DRL agent saves the experience $e(t)$ in the experience replay memory.
    The DRL agent selects an action $a(t)$ based on the ϵ-greedy algorithm: with probability $1-\epsilon$, it selects the action $a(t) = \arg\max_{a \in \mathbf{A}} Q_{eval}(s(t), a; w)$; with probability ϵ, it randomly selects an action from the action space **A**.
    The DRL agent chooses at random a mini-batch with Z experience sample from the experience replay memory.
    The DRL agent inputs the experiences into the action selection module for batch-training to updates the weights of the evaluation network w.
    In every M time slots, updates the weights of the target network with $w^- = w$.
    if Q_{eval} is converged **then**
        $Q_{targ} = Q_{eval}$.
        Break
    end if
end for

The goal of the ϵ-greedy algorithm is to balance experience exploitation and exploration of the optimum actions [32]. Specifically, since the number of experience samples stored in the local memory is seldom in the early stage of the training process, the DRL agent should spend most of the time on the exploration of the best policy. With the accumulation of experience samples, the DRL agent should increase the time fraction spent on the exploitation of the obtained experience. As a result, the parameter ϵ is updated as follows

$$\epsilon(t+1) = \max\{\epsilon_{\min}, \epsilon(t) - (\epsilon_0 - \epsilon_{\min})/\alpha_\epsilon\}, \tag{14}$$

where ϵ_0 is the initial value, $\epsilon_{\min}$ is a lower bound of ϵ , and α_ϵ is the attenuation factor.

Next, we discuss the complexity of the proposed algorithm. The number of multiplications in IDS is given by $D \doteq \tilde{K}d_1 + \sum_{g=1}^{G-2} d_g d_{g+1} + (\tilde{e}+\tilde{t})d_{G-1}$, where G is the number of layers in network, $\tilde{K}$ is the size of the input layer which is proportional to the number of users, d_g is the number of units in the gth full connected layer, and $\tilde{e}$ and $\tilde{t}$ are the numbers of units in the evaluation network and target network, respectively. Therefore, the computational complexity is given by $O(D)$ at each time step of IDS.

4.4 Adaptive IDS for dynamic network environment

In practice, the network environment could be dynamic, meaning that certain parameters, such as aggregate channel parameters p, may change over time due to factors such as user mobility. An ideal scheduling policy should be able to adapt to the network dynamics. In this subsection, we will explore the design of adaptive scheduling policies for our problem.

Let us assume that there are an infinite number of data blocks to be disseminated to the users. These data blocks are transmitted sequentially, i.e., once a data block is successfully disseminated, the buffer contents of the RS and all users will be removed, and the transmission of the next data block begins. To detect network dynamics, we employ the following method: Let D^* denote the average dissemination delay of the scheduling policy learned by the IDS algorithm, and let D_i denote the dissemination delay of the ith data block. Once a data block i is successfully transmitted, we calculate the average dissemination delay of the previous L data blocks, i.e.,

$$\frac{1}{L}\sum_{j=i-L+1}^{i} D_j. \tag{15}$$

If Eq. (15) is greater than $D^*(1+\xi)$ for some constant ξ, the IDS algorithm will be used for a new round of training based on neural network parameters to adjust the scheduling policy to network dynamics adaptively. Since we use the average dissemination delay of L data blocks, i.e., Eq. (15), to detect network dynamics, the fluctuation in the dissemination delay of a data block is reduced, and we can set ξ as a small number, e.g., 0.1. The details of the adaptive IDS algorithm are shown in Algorithm 2. The performance of the adaptive IDS algorithm will be evaluated through simulations in the following section.

Algorithm 2 Adaptive IDS

RS performs IDS to learn a scheduling policy.
Compute the average dissemination delay D^* of the learned policy.
for $i = 1, 2, 3...$ **do**
  Transmission of ith data block using the learned policy.
  Record the dissemination delay D_i.
  if $\frac{1}{L}\sum_{j=i-L+1,\ldots,i} D_j > D^*(1+\xi)$ **then**
    Perform a new round of training using IDS, and update D^*.
  end if
end for

5 Results and discussion

In this section, we will evaluate the performance of our IDS by simulation. We will compare the performance of our algorithms to the following baseline schemes.

(i) ARQ

This is a traditional transmission scheme without employing rateless code. An original packet is fetched from the BS by the RS and is then broadcast to the users until it has been received by every user. The retransmissions of the lost packet are triggered by ACK/NAK feedbacks which are assumed to be free and perfect. The above procedure will be repeated for K times so that all users receive the data block successfully.

(ii) GREEDY

Whenever possible, the RS greedily broadcasts a packet to the users; That is, the RS chooses to fetch a packet when its buffer is empty or when all of the user channels are in the OFF state.

(iii) WBP-TRACK [11]

This is a state-of-the-art scheme for file dissemination in two-hop half-duplex relay networks using rateless code. At each time slot, the RS chooses to fetch a coded packet from the BS if the overall quality of the broadcast channel between the RS and the users is not 'high' enough, and broadcasts a buffered packet to the users otherwise.

Once all users have received the K coded packets and successfully decoded the original data block, the transmission is considered complete. However, WBP-TRACK relies on a perfect BS-to-RS channel, i.e., it can only function when $p_0 = 1$ and cannot easily be extended to the general case that p_0 falls within the range (0, 1).

In our simulation, we also include a lower bound of the dissemination delay proposed in [11], which is obtained by numerically solving the fluid approximation [33] of our problem.

We use the open-source framework PyTorch [34] for the implementation of the IDS algorithm and use MATLAB for performance comparison of all the algorithms. The simulation runs on a PC platform equipped with Windows 10 operating systems, 16G RAM, Intel i5 4.10GHz CPU and Nvidia GTX 1080 Ti GPU. The Dueling-DDQN is composed of an input layer with $2N + 2$ ports, corresponding to $2N + 2$ elements in $s(t)$, two fully

connected hidden layers, then connected to the state value function network and advantage function network with 50 neurons, respectively, and an output layer of two ports, which correspond to two candidate actions available to the RS. One hundred neurons with the ReLU activation function are present in each hidden layer. Thus, the number of multiplications is $(2N + 2) \times 100 + 100 \times 100 + 100 \times 50 \times 2$.

We use an ϵ-greedy algorithm with $\epsilon_0 = 0.1, \epsilon_{\min} = 0.001$ and $\alpha_\epsilon = 2000$. The parameters of the proposed algorithms are set as $Z = 30, T = 128, M = 10$, reward discount rate $\gamma = 0.99$ and experience replay memory size is 10^4. The Adam optimizer with a learning rate 10^{-4} is used for weights updating. The size of the block is set to $K = 100$. All results are averaged over 10^4 random runs. A summary of the major parameters used in our simulation is shown in Table 2.

Figure 4 presents the dissemination delay of the trained policy using IDS and DQN as a function of the number of iterations when $N = 2$ and $p = (1, 0.5, 0.9)$. The dissemination delay decreases as the number of training iterations increases for both methods, and IDS has a faster convergence speed compared with DQN.

Figure 5 shows the average dissemination delay performance for symmetric-user systems with $N = 2$, $p_0 = 1$, and $p_1 = p_2 = q$, where the value of q is varied. The figure shows that, as the channel conditions improve, i.e., as q increases, the dissemination delay of each scheme decreases. IDS performs the best, incurring a dissemination delay that is only 2–5% higher than the lower bound, and it reduces the dissemination delay by up to 7%, 23%, and 36% compared with WBP-TRACK, ARQ, and GREEDY, respectively.

The dissemination delay performance for asymmetric-user systems with varying N is presented in Fig. 6. For each N, we set the aggregate channel parameters as $p_i = 0.5 + 0.4(i - 1)/(N - i)$ for all $i \in I$. The performance of GREEDY is almost constant in this scenario, since it mainly depends on the channel parameter of the "worst" user, i.e., user 1 with $p_1 = 0.3$. The performance of ARQ decays rapidly with N, and one can expect that GREEDY would be outperformed by ARQ as N increases. IDS performs closest to the lower bound and reduces the dissemination delay by 4% compared to WBP-TRACK.

Table 2 List of parameters

Parameters	Values
K	100
T	128
Z	32
M	10
ϵ_0	0.1
$\epsilon_{\min}$	0.001
α_ϵ	2000
γ	0.99
Optimizer	Adam
Activation Function	ReLU
Learning rate	10^{-4}
Experience replay memory size	10^4

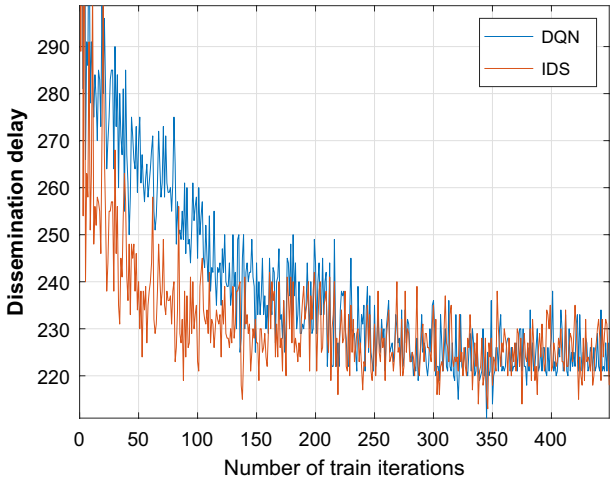

Fig. 4 Dissemination delay of the trained policy using IDS and DQN

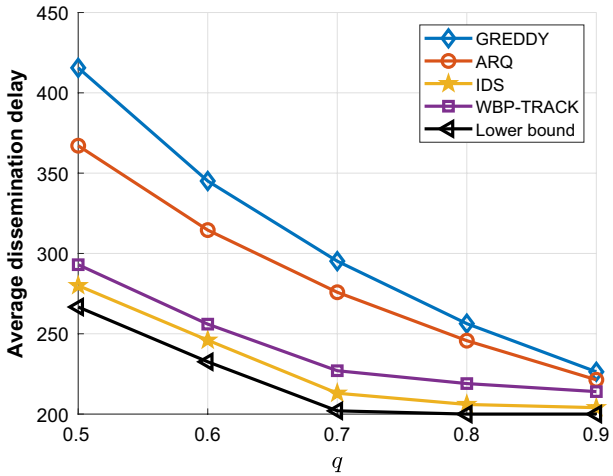

Fig. 5 Symmetric-user systems with $N = 2$, $p_0 = 1$, and $p_i = q$ for all $i \in I$ with varying q

The previous simulations focus on systems with a perfect BS-to-RS channel. Next, we will compare the dissemination delay performance for systems with an imperfect BS-to-RS channel, and WBP-TRACK will not be included in the following simulations.

The average dissemination delay performance for symmetric-user systems with $N = 2$, $p_0 = 0.5$, and $p_1 = p_2 = q$ with varying q is displayed in Fig. 7. IDS significantly reduces dissemination delay, where the gains are up to 22% and 40% compared with ARQ and GREEDY, respectively. Figure 8 considers the case that $N = 2, p_i = 0.7$ for all $i \in I$ and varying p_0. The performance of all schemes improves as the quality of the BS-to-RS channel increases. GREEDY incurs 23–32% higher dissemination delay than IDS, and ARQ achieves 16–22% higher delay than IDS.

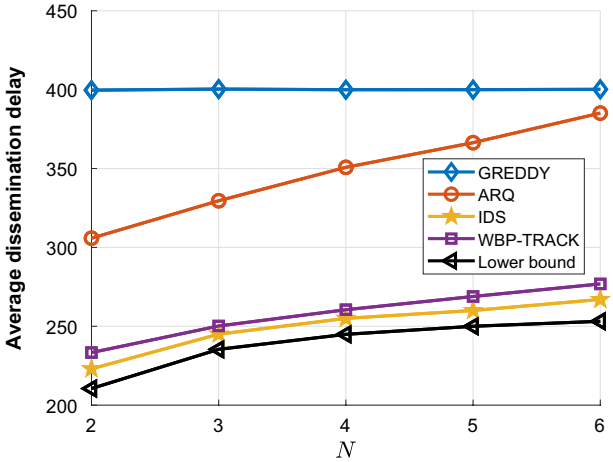

Fig. 6 Asymmetric-user systems with $p_0 = 1, p_i = 0.5 + 0.4(i-1)/(N-1)$ for all $i \in I$ and varying N

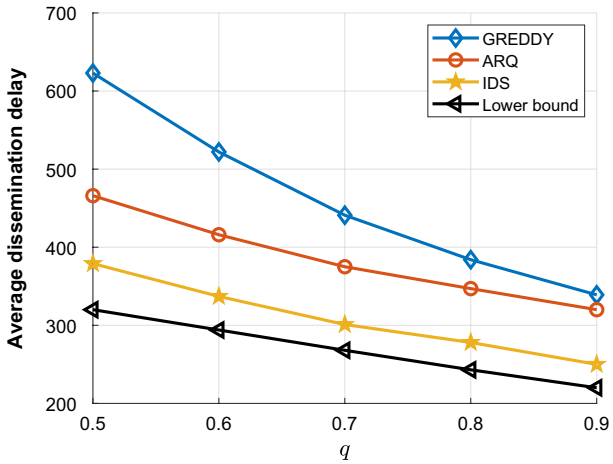

Fig. 7 Symmetric-user systems with $N = 2, p_0 = 0.5$, and $p_i = q$ for all $i \in I$ with varying q

With $p_0 = 0.7$ and $p_i = 0.5 + 0.4(i-1)/(N-1)$ for all $i \in I$, Fig. 9 shows how the dissemination delay changes varies with N for asymmetric-user systems. The dissemination delay incurred by IDS is within 10% of the lower bound. The dissemination delays for GREEDY and ARQ are up to 24% and 40% higher than that of IDS, respectively.

Next, we evaluate the performance of the adaptive IDS in Fig. 10, we set $N = 2, L = 20$ and $\xi = 0.1$. The aggregate channel parameters are set as $p = (1, 0.7, 0.7)$ initially, such that the dissemination delay of IDS is around 230 for each data block. At the beginning of the transmission of the 200th data block, the aggregate channel parameters are reduced to $p = (1, 0.5, 0.5)$. We observe that the dissemination delay of the proposed algorithm quickly increases, since the previous learned policy is no longer efficient in the changed network environment. However, our adaptive IDS is able to detect the network dynamics by measuring the average dissemination delay of the previous L data

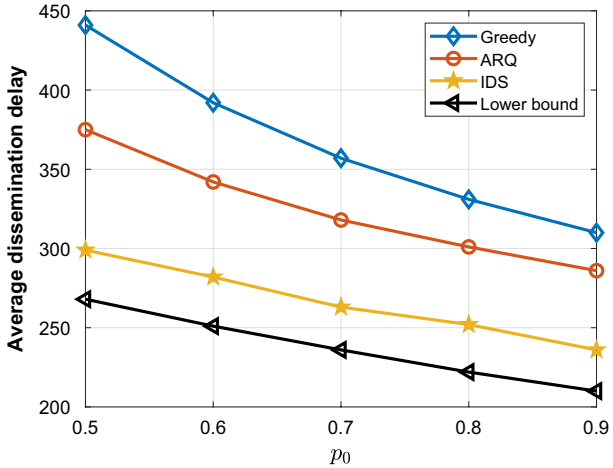

Fig. 8 Symmetric-user systems with $N = 2, p_i = 0.7$ for all $i \in I$ and varying p_0

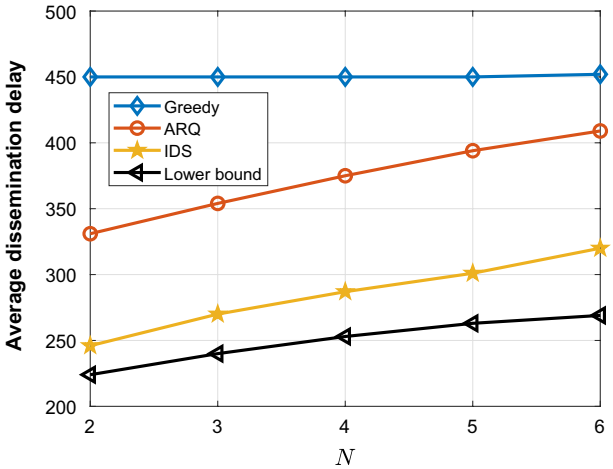

Fig. 9 Asymmetric-user systems with $p_0 = 0.8, p_i = 0.5 + 0.4(i-1)/(N-1)$ for all $i \in I$ with varying N

blocks, i.e., (15) in our paper, and comparing it with the average dissemination delay of the learned algorithm. If an abnormal change is detected, it will perform a new round of training based on the previously learned parameters to update the scheduling policy. Thus, in Fig. 10, we can observe that the dissemination delay of our algorithm reduces from 450 to 300 gradually from the 200th data block to 220th data block. That is, our algorithm automatically adjusts with changes in the environment.

6 Conclusions

In this paper, we examined the minimum-delay file dissemination problem using rateless code in two-hop relay-based HetNets. We constructed the MDP formulation of the scheduling problem at the half-duplex RS. Due to the large state of the MDP system, we proposed an intelligent DRL-based scheduling algorithm. We also extended the

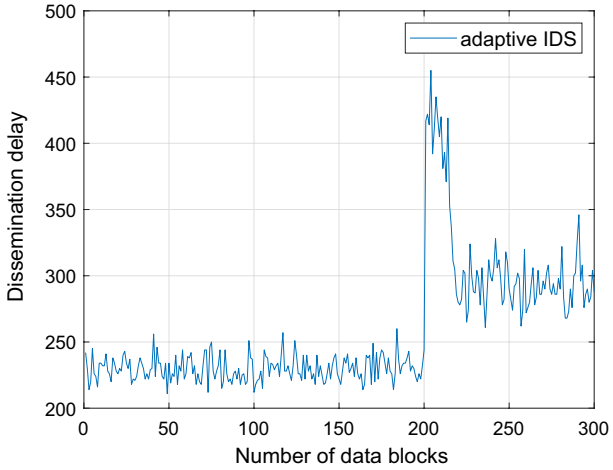

Fig. 10 The dissemination delay in dynamic network environment

proposed algorithm to adapt to dynamic network conditions. Simulation results demonstrated that the proposed algorithm outperforms other transmission schemes and performs nearly as well as a lower bound of the dissemination delay.

Abbreviations

5G	Fifth generation
6G	Sixth generation
ARQ	Automatic repeat-reQuest
BS	Base station
BSI	Buffer state information
CSI	Channel state information
DQN	Deep Q-network
DDQN	Double deep Q-network
DRL	Deep reinforcement learning
HetNets	Heterogeneous networks
MCS	Modulation and coding scheme
MDP	Markov decision process
QoS	Quality of service
RB	Resource block
RL	Reinforcement learning
RS	Relay station

Acknowledgements
Not applicable.

Author contributions
CC and CL conceived the original idea and completed the theoretical analysis. XY and BM improved the algorithm of the article. ZW carried out the experiments and data analysis. All authors provided useful discussions and reviewed the manuscript. All authors read and approved the final manuscript.

Funding
This paper was supported in part by the National Natural Science Foundation of China (62111540270, 61871468), the Zhejiang Provincial Natural Science Foundation of China (LZ23F010003, LQ23F010009), the Fundamental Research Funds for the Provincial Universities of Zhejiang (XRK22005), the Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (2013E10012), and the Zhejiang Gongshang University "Digital+" Disciplinary Construction Management Project (SZJ2022B010).

Availability of data and materials
The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 4 April 2023 Accepted: 17 November 2023
Published online: 28 November 2023

References

1. Y. Xu, G. Gui, H. Gacanin, F. Adachi, A survey on resource allocation for 5g heterogeneous networks: current research, future trends, and challenges. IEEE Commun. Surv. Tutor. **23**(2), 668–695 (2021)
2. J.J. Gimenez, J.L. Carcel, M. Fuentes, E. Garro, S. Elliott, D. Vargas, C. Menzel, D. Gomez-Barquero, 5g new radio for terrestrial broadcast: a forward-looking approach for NR-MBMS. IEEE Trans. Broadcast. **65**(2), 356–368 (2019)
3. E. Garro, M. Fuentes, J.L. Carcel, H. Chen, D. Mi, F. Tesema, J.J. Gimenez, D. Gomez-Barquero, 5g mixed mode: Nr multicast-broadcast services. IEEE Trans. Broadcast. **66**(2), 390–403 (2020)
4. M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, M. Zorzi, Toward 6g networks: use cases and technologies. IEEE Commun. Mag. **58**(3), 55–61 (2020)
5. C. Sexton, N.J. Kaminski, J.M. Marquez-Barja, N. Marchetti, L.A. DaSilva, 5g: adaptable networks enabled by versatile radio access technologies. IEEE Commun. Surv. Tutor. **19**(2), 688–720 (2017)
6. D.J. MacKay, Fountain codes. IEE Proc. Commun. **152**(6), 1062–1068 (2005)
7. B.T. Swapna, A. Eryilmaz, N.B. Shroff, Throughput-delay analysis of random linear network coding for wireless broadcasting. IEEE Trans. Inf. Theory **59**(10), 6328–6341 (2013)
8. G.H. Sim, J. Widmer, B. Rengarajan, Opportunistic finite horizon multicasting of erasure-coded data. IEEE Trans. Mob. Comput. **15**(3), 705–718 (2016)
9. T. Low, M. Pun, Y.P. Hong, C.J. Kuo, Optimized opportunistic multicast scheduling (oms) over wireless cellular networks. IEEE Trans. Wirel. Commun. **9**(2), 791–801 (2010)
10. H. Khamfroush, D.E. Lucani, J. Barros, Network coding for wireless cooperative networks: simple rules, near-optimal delay. In: 2014 IEEE International Conference on Communications Workshops (ICC), pp. 255–260 (2014)
11. C. Chen, S.J. Baek, Multicast scheduling for relay-based heterogeneous networks using rateless codes. IEEE Trans. Mob. Comput. **16**(11), 3142–3155 (2017)
12. A. Alwarafy, M. Abdallah, B.S. Ciftler, A. Al-Fuqaha, M. Hamdi, The frontiers of deep reinforcement learning for resource management in future wireless hetnets: techniques, challenges, and research directions. IEEE Open J. Commun. Soc. **3**(1), 322–365 (2022)
13. G.P. Koudouridis, Q. He, G. Dán, An architecture and performance evaluation framework for artificial intelligence solutions in beyond 5g radio access networks. EURASIP J. Wirel. Commun. Netw. **2022**(1), 1–32 (2022)
14. M.U. Iqbal, E.A. Ansari, S. Akhtar, Interference mitigation in hetnets to improve the qos using q-learning. IEEE Access **9**(1), 32405–32424 (2021)
15. M. Simsek, M. Bennis, I. Güvenç, Context-aware mobility management in hetnets: a reinforcement learning approach. In: 2015 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1536–1541 (2015)
16. X. Wang, H. Chen, S. Li, A reinforcement learning-based sleep scheduling algorithm for compressive data gathering in wireless sensor networks. EURASIP J. Wirel. Commun. Netw. **2023**(1), 28 (2023)
17. W. AlSobhi, A.H. Aghvami, Qos-aware resource allocation of two-tier hetnet: a q-learning approach. In: 2019 26th International Conference on Telecommunications (ICT), pp. 330–334 (2019)
18. V. Raj, I. Dias, T. Tholeti, S. Kalyani, Spectrum access in cognitive radio using a two-stage reinforcement learning approach. IEEE J. Sel. Top. Signal Process. **12**(1), 20–34 (2018)
19. F. Tang, Y. Zhou, N. Kato, Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5g hetnet. IEEE J. Sel. Areas Commun. **38**(12), 2773–2782 (2020)
20. Q. Su, B. Li, C. Wang, C. Qin, W. Wang, A power allocation scheme based on deep reinforcement learning in hetnets. In: 2020 International Conference on Computing, Networking and Communications (ICNC), pp. 245–250 (2020)
21. F. Yang, W. Wu, X. Wang, Y. Zhang, P. Si, Deep reinforcement learning based handoff algorithm in end-to-end network slicing enabling hetnets. In: 2021 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–7 (2021)
22. M. Ma, V.W.S. Wong, A deep reinforcement learning approach for dynamic contents caching in hetnets. In: ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–6 (2020)
23. L. Li, C.F. Kwong, Q. Liu, J. Wang, A smart cache content update policy based on deep reinforcement learning. Wirel. Commun. Mob. Comput. **2020**(1), 11 (2020)
24. Q. Cong, W. Lang, Double deep recurrent reinforcement learning for centralized dynamic multichannel access. Wirel. Commun. Mob. Comput. **2021**(1), 10 (2021)
25. Y. Geng, E. Liu, R. Wang, Y. Liu, Hierarchical reinforcement learning for relay selection and power optimization in two-hop cooperative relay network. IEEE Trans. Commun. **70**(1), 171–184 (2021)
26. R. Cogill, B. Shrader, A. Ephremides, Stable throughput for multicast with inter-session network coding. In: MILCOM 2008–2008 IEEE Military Communications Conference, pp. 1–7 (2008)
27. S. Wang, H. Liu, P.H. Gomes, B. Krishnamachari, Deep reinforcement learning for dynamic multichannel access in wireless networks. IEEE Trans. Cogn. Commun. Netw. **4**(2), 257–265 (2018)
28. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
29. N.C. Luong, D.T. Hoang, S. Gong, D. Niyato, P. Wang, Y.C. Liang, D.I. Kim, Applications of deep reinforcement learning in communications and networking: a survey. IEEE Commun. Surv. Tutor. **21**(4), 3133–3174 (2019)

30. H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, pp. 1–7 (2016)
31. Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling network architectures for deep reinforcement learning. In: Proceedings of The 33rd International Conference on Machine Learning, vol. 48, pp. 1995–2003 (2016)
32. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, 2018)
33. S. Meyn, *Control Techniques for Complex Networks* (Cambridge University Press, Cambridge, 2008)
34. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., Pytorch: an imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32**(1), 1–12 (2019)

Publisher's Note