RL-based Resource Allocation in mmWave 5G IAB Networks

Bibo Zhang, Francesco Devoti, Ilario Filippini
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy
name.surname@polimi.it

***Abstract*—5G standardization has envisioned mmWave communications as a promising direction to expand the capacity of current mobile radio networks. However, communications at high frequency are characterized by extremely harsh propagation conditions, thus requiring a high base station deployment density. To solve this issue, from both technical and economic perspective, 3GPP has proposed mmWave access networks based on an Integrated Access and Backhaul (IAB) multi-hop architecture.**

IAB networks require fine-tuning of the available resources in a complex setting, due to directional transmissions, device heterogeneity, and harsh propagation conditions. The latter, in particular, characterize the operations of such networks, resulting in links with very different levels of availability. For this reason, traditional optimization techniques do not provide the best performance in these conditions. We believe, instead, Reinforcement Learning (RL) techniques can implicitly consider the dynamics of the network links and learn the best resource allocation strategy in networks with intermittent links. In this paper, we propose an RL-based resource allocation approach that shows the advantages of these techniques in dynamic environmental conditions.

I. Introduction

5G standardization process has resorted to high-frequencies communications between 25 GHz and 50 GHz, also named millimeter waves (mmWaves), as one of the main reliefs from the global mobile traffic growth which is challenging the capacity of communication technologies below 6 GHz. The availability of several GHz of bandwidths at those frequencies can potentially provide tens of Gbps for the mobile access.

However, this attractive advantage comes at the cost of a harsh propagation environment: very high path losses and no propagation through obstacles (not only vehicle and buildings, but also human bodies). The use of sophisticated phased antenna arrays has given a big help in addressing these issues. Nevertheless, that is not sufficient. MmWave deployments are coverage-limited, thus 5G mmWave access networks require a denser base station placement than traditional mobile radio networks. This translates into high installation costs for operators that need to backhaul many sites with fibers.

In order to provide a technically and economically viable solution to the network densification cost, 3GPP standardization body has proposed a new multi-hop access architecture, named Integrated Access and Backhaul (IAB)[1]. The idea is to place some smaller and simpler relay nodes, called IAB nodes, in the coverage area of the main mmWave base station (BS) in order to forward mmWave BS's data packets to users (UEs). The mmWave BS, called Donor gNB (DgNB), is in charge of managing the resources allocation in the network of associated IAB nodes. In particular, since the proposed MAC solution is based on TDMA, it involves the optimization of the routing paths and the scheduling of directional transmissions along established links.

Routing and scheduling in wireless multi-hop networks are typically carried out via optimization techniques considering all available links [2–4]. However, the harsh propagation environment at mmWave frequencies and the strong impact of the obstacles on link availability make these approaches inadequate for mmWave IAB networks. Indeed, the possibility of using a link essential to provide a good performance can be hindered by its unpredictable on-off behavior, thus destroying the advantages of the optimization. The optimization could in principle be performed each time the network undergoes a change. However, optimization algorithms are usually time consuming, which makes this solution infeasible.

We believe that resorting to Reinforcement Learning (RL) techniques can be a promising solution as the intrinsic adaptability of these algorithms to the environment conditions can solve the above-mentioned issues. Indeed, RL agents can be trained to play against the environment to understand what the best strategy is, even when the environment's reply is stochastic. We can perform offline training through a realistic instance of the actual environment, or we can have an online training and operating system that learns while sending packets through IAB nodes to UEs.

In this paper, we introduce an RL-based approach to tackle flow allocation and link scheduling in mmWave IAB networks, jointly coordinating access and backhaul parts to maximize throughput in a multi-hop network architecture. We place emphasis on more realistic environments and unreliable networks which are vulnerable to dynamic and complicated blockages. We propose an offline and an online version of the approach, which we evaluate against traditional approaches via numerical simulations.

The rest of the paper is organized as follows. We first discuss related works in Section II, then we provide a system overview and a detailed formulation of the problem in Section III. Our approach is detailed in Section IV, while the results of the numerical analysis are showcased and discussed in Section V. Finally, Section VI concludes the paper with some final remarks.

II. Related Work

The problem of resource management in mmWave networks has been investigated in recent literature, which can be divided into several categories with respect to network scenarios, controllable network variables, optimization targets and solutions.

Considering network scenarios, only recent works focus on IAB networks [5, 6]. Some work [7–9] take into account possible blockages in connections, for instance, static blockages for specific propagation scenarios [7], urban buildings [8], concise dynamic blockages in backhaul [9]. The blockage assumptions above are, in some sense, limited when solving complicated blockages (e.g., pedestrian and vehicle traffic).

In terms of controllable variables in mm-Wave networks, current works pay more attention to: routing and scheduling [2, 5, 8, 10–13], bandwidth and beamwidth assignment [7, 14], frame / slots reconfiguration [2, 11, 15] and network formulation [7]. As for targets, plenty of works intend to maximize throughput [2, 6–9, 11, 12, 14, 15], while some other works prefer to minimize transmission delay [12, 13] and consumed energy [10].

Several approaches have been introduced to solve resource allocation issues. Maximum independent set (MIS) based heuristic algorithms are provided in [11], while some other heuristic methods are proposed in [8, 15, 16].

Apart from the optimization based approaches, RL based approaches have been raised to solve various problems. [13] proposes a semi-distributed multi-armed bandit based learning algorithm to minimize the end-to-end latency, which is proven to be adaptive to load imbalance, channel variations and link failures. [6] resorts to regret RL to complete route selection and tackles the problem of rate allocation by successive convex approximation method. [14] maximizes data rate by controlling transmitter beamwidth and transmission power allocation using risk sensitive RL. [9] presents a DRL approach to assign backhaul resources to users.

However, none of them considers mmWave IAB scenarios, and, in particular, the problem of sending data packets over a multi-hop mmWave network with intermittent links.

III. System Overview and Problem Formulation

A. Network Scenario, Link Patterns and Blockages

An IAB scenario typically consists of a multi-hop wireless network with one Donor gNB (DgNB), a set of relay nodes (IAB nodes / RNs) and users (UEs) which can reach DgNB via either direct links or multi-hop IAB nodes. The DgNB is connected with the core network via a high capacity link (i.e., fiber). We consider here the Fixed Wireless Access (FWA) use case, which is expected to be the first application of mmWave IAB networks. UEs are nodes placed on rooftops and balconies to provide connectivity to home customer networks.

The *backhaul links* between DgNB and IAB nodes or IAB nodes and IAB nodes, and *access links* between DgNB and UEs or IAB nodes and UEs, are wireless and share the same mmWave frequency bands (i.e., in-band backhaul). We represent the network architecture as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ includes nodes (i.e., DgNB, IAB nodes and UEs) and $\mathcal{E}$ involves possible links among nodes in $\mathcal{V}$. We use $\mathcal{R}, \mathcal{U} \subset \mathcal{V}$ to denote the sets of IAB nodes and UEs,

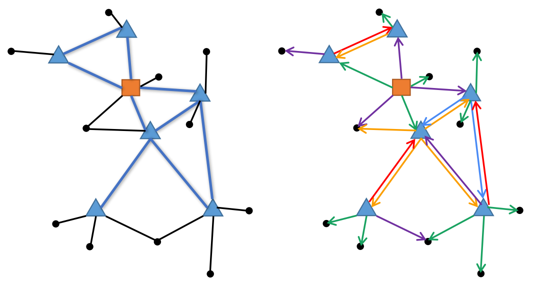


Fig. 1: A toy example of network and pattern construction. The square, triangles and black points are DgNB, IAB nodes and UEs. The figure on the left shows the network topology: thick blue lines are backhaul links while the others are access links. Five example patterns are shown in different colors in the figure on the right.

respectively. If not specified, DgNB node is also included in IAB node set as a special node. We focus on downlink transmission, taking care of total amount of data passed from DgNB to UEs.

We consider a frame of $T \in \mathbb{N}$ slots with equal duration δ, in a time domain $\mathcal{T}$. Basic TDMA scheme assumes each slot exclusively occupied by a single link, which can hinder the performance of highly spacial-differential mmWave signals. Compared with TDMA, we consider a space-division multiple access (SDMA) system to make good use of high directivity of mmWave antennas, allowing multiple concurrent transmissions in each slot. We refer to a set of links activated in parallel as a *pattern* that meets the channel conditions (e.g., interference requirements, antenna patterns), half-duplex and multi-beams limits. Figure 1 shows a toy example of a network graph model and its corresponding pattern construction. Note that a pattern may include both access and backhaul links. In each slot, a pattern is selected among the set of those available and the associated link transmissions are activated. The optimal sequence of patterns allows to maximize the number of bits transferred from DgNB to UE, which corresponds to the maximization of downlink throughput in IAB networks.

We also introduce dynamic blockages to our system. Both backhaul and access links are exposed to blockages. Link blockages vary in a small-grained scale (e.g., slot by slot). In each slot, each link, with a certain probability p_b, is blocked, and with the probability $1 - p_b$ stays on. When a link is blocked, no data can be transferred. Links with different reliability are characterized by different values of p_b. We use different distribution of link blockage probabilities to take into account environmental changes.

To face up to environments with blockages, RL is the most promising approach as it can be adaptive to system dynamics. Indeed, traditional optimization algorithms rely on static link availability that may not be met during real time operations.

B. Problem Formulation

We intend to jointly address flow allocation and link scheduling to maximize the IAB network throughput, taking care of UE fairness. Several realistic constraints need to be considered, such as half-duplex, simultaneous multi-beams, interference requirements, power limitations. However, relying on a pattern formulation, all these issues are confined to the generation of feasible patterns, which is discussed in Section IV-A. Thanks to this separation, we can focus on flow allocation and pattern scheduling for throughput maximization. We formulate the problem as follows.

Our target is to maximize downlink cell throughput (i.e., the total flow of bits obtained by UEs per frame):

$$\max \sum_{r \in \mathcal{R}, u \in \mathcal{U}} b_{r,u}, \tag{1}$$

where $b_{r,u}$ is a variable representing the flow of bits transferred over the access link (r, u) during the overall frame.

1) UE Fairness: A min-rate constraint is set to guarantee the fairness among users instead of maximizing the throughput by prioritizing some well-positioned UEs, where ρ_{min} is the minimum data rate of every serviceable UE:

$$\sum_{r \in \mathcal{R}} b_{r,u} \geqslant \rho_{min} \cdot T \cdot \delta, \ \forall u \in \mathcal{U}. \tag{2}$$

2) Flow Allocation: Traffic originates from the core network and goes to UEs via DgNB and IAB nodes. The incoming traffic to IAB nodes (DgNB and RNs) must equalize to the outgoing traffic in a frame, where b_{core} is the traffic sent from the core network and entering DgNB only.

$$\sum_{n \in \mathcal{R}, n \neq r} b_{n,r} + b_{core} = \sum_{v \in \mathcal{R} \cup \mathcal{U}, v \neq r} b_{r,v}, \ \forall r \in \mathcal{R}. \tag{3}$$

3) Pattern Scheduling: The overall capacity provided by link activation in a frame must support the flow allocated. Link transmissions in scheduled patterns create a pipe (capacity) where flows of bits can be accommodated. This rationale is expressed below, where $c^i_{n,m}$ is the capacity of link (n, m) compatible with SINR values when pattern i in pattern set $\mathcal{P}$ is activated and x^t_i is a binary decision variable controlling whether pattern i should be activated in slot t.

$$\sum_{t \in \mathcal{T}, i \in \mathcal{P}} (c^i_{n,m} \delta) x^t_i \geqslant b_{n,m}, \ \forall n \in \mathcal{R}, m \in \mathcal{R} \cup \mathcal{V}. \tag{4}$$

Solving the optimization problem described by the above formulation provides an optimal pattern sequence, through x^t_i variables, for the flow and scheduling management in static IAB scenario. Indeed, the optimization assumes all links are always available, and $c^i_{n,m}$ is fully guaranteed when pattern i is selected. However, this assumption may lead to bad performance when some links encounter static or even dynamic blockages. This is the reason why we propose an approach based on RL, which can automatically adapt the best scheduling policy to a time-varying environment.

C. Deep Reinforcement Learning

Reinforcement Learning (RL) is based on an agent that executes a sequence of interactions with an environment, which follows an MDP (Markov Decision Process), to gradually accumulate experience. In particular, at time step t, the environment is in state s_t. Based on this state, the agent chooses an action a_t to perform on the environment. At time step $t + 1$, the environment transits to the next state s_{t+1} with some probability and gives a reward r_t back to the agent. The agent's goal is to maximize the long-term cumulative reward which is the *expected return* $\mathbb{E}_\pi[G_t] = \mathbb{E}[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k]$. T is the last step in an interaction episode and γ is a discount factor controlling the importance of future reward to current concern. There exist two categories of RL algorithms: value function based and policy gradient.

Value-function-based methods choose actions based on a value function estimated at each state. In particular, the state value function $v_\pi(s_t) = \mathbb{E}_\pi[G_t|s_t]$ is the expected return from state s when the agent follows the policy π. The action-value function $q_\pi(s_t, a_t) = \mathbb{E}_\pi[G_t|s_t, a_t]$ is the expected return from state s, taking action a and following policy π afterwards. The best policy π^*, which provides the best actions' strategy at each state, is chosen considering these functions and the impact of chosen actions on the expected reward. Value functions can be written in tabular form or being approximated with other functions, such as linear functions, kernels, or deep neural networks (DNNs). Deep RL employs DNNs as approximation.

Policy gradient approaches directly represent policy as a probability function of taking an action at a state with the vector of parameters θ, i.e., $\pi(a_t|s_t, \theta) = Pr\{a_t|s_t, \theta\}$. Like the value function above, $\pi(a_t|s_t, \theta)$ could also be represented by a DNN where θ stores the connection weights of the network. Parameter θ is updated according to the value function gradient, in the direction of increasing the value determined by the policy.

The actor critic method derives from policy gradient and incorporates the strength of value based methods. Different from policy gradient, actor critic uses both a probabilistic policy function and a state value function. This allows to further drive the system towards policy functions that provide better values for system states. The approach we proposed is based on Advantage Actor Critic (A2C), described in details in Section IV.

IV. Resource Allocation in MmWave IAB Networks with Deep RL

We propose an A2C-based approach which can face complicated IAB scenario with dynamic blockages. We separate the problem into two parts: 1) pattern generation, considering hardware constraints, and 2) flow allocation and pattern scheduling, in which an RL agent activates sequences of per-slot patterns to transfer flows from DgNB to UEs.

A. Link Patterns Generation

Optimally solving the formulation presented in Section III-B requires us to provide as input the whole set of possible patterns that can be activated in the network. However, this set has a cardinality that increases exponentially with the number of links, thus creating a formulation with a huge number of binary variables x^t_i, making the problem intractable.

In order to solve this issue, the technique of Column Generation (CG) can be applied. It consists in solving a linear relaxation of the formulation in Section III-B (the so-called Master Problem) with an initial set of patterns and, using dual variables related to constraints (4) to find out if an additional pattern can potentially improve the value of the objective function. This additional pattern can be found by solving a problem (the so-called Pricing Problem) in which the best links indicated by dual variables are activated according to physical and hardware constraints. Pricing Problem includes all technological aspects we need to consider: channel model, SINR values required to activate specific Modulation and Coding Schemes (MCSs), availability of power control to reduce the interference impact, half-duplex constraints, etc. These captured technological aspects also include how devices are engineered such as the number of bidimensional antenna panels and the number of beams that can be simultaneously activated by the panel in the pattern (i.e., time slot). The output of the Pricing Problem is a *pattern*: a compatible set of links that can be activated together without violating all the above mentioned constraints.

Every time a new objective-improving pattern is found, it is included in the set of available patterns and a new linearly-relaxed Master Problem is solved, and the process iterates. The addition of new patterns stops when no improving pattern can be generated without violating Pricing Problem's constraints. Once the pattern generation has terminated, the integer Master Problem is solved with the final set of patterns. The CG approach provides results very close to the optimum.

In our RL-based approach we leverage CG as a tool to generate a good set of patterns from which the RL agent can choose from. We have implemented a CG algorithm by formulating Master and Pricing models and collecting all the patterns included in the last iteration before the generation stops. Since this is not the main focus of this paper and it requires a long description of the models and the constraints used, we refer to similar approaches in literature, which optimize routing and scheduling in multi-hop wireless networks using CG approaches [3, 4].

CG provides a set of good patterns to transfer data from DgNB to UEs in the case of always-available links. The RL agent will select among these patterns the best combination to address the unreliability of some links. The generation of the pattern set is a procedure that can be run only once, at the beginning of the network operations.

To the purpose of our RL-based approach, we do not need to solve the final integer Master Problem, which is a time consuming task. However, we did it in order to compute the quasi-optimal pattern sequence selected in ideal link conditions to be used as a benchmark to compare against our RL-based approach.

B. Buckets-Pipes Game Formulation

In order to perform flow allocation and pattern scheduling slot by slot in the RL environment, we reformulate the system model as a game the RL agent has to deal with. DgNB, IAB nodes, and UEs are regarded as buckets that store data bits as water. Links act as pipes connecting these buckets. Link blockage is similar to pipe congestion which determines the unavailability of links. The pattern activated in each slot controls which group of pipes' valves to be opened. The target is to maximize the total amount of water in UEs' buckets in a frame.

Accordingly, the optimization objective (1) serves as the long-term expected return RL maximizes. Flow allocation, pattern scheduling and UE fairness depend on which patterns the RL agent selects to activate and how the traffic flows through the system. RL agent chooses one pattern of links (pipes) to activate in each slot based on the action probability given by the policy. The flow transmission obeys the following rules. Data traffic can be buffered in the queues at IAB nodes[1]. The total number of bits transferred from an IAB node over outgoing links in one slot is limited by the amount of bits in the queue, as indicated by flow balance equations (3). The maximum number of bits each link can transmit is limited by the link capacity allowed by the activated pattern, which refers to pattern scheduling constraints (4). UE Fairness (2) is achieved by equally sharing the amount of buffered data transmitted among activated outgoing links of each single IAB node.

C. A2C Based Flow Allocation and Pattern Scheduling

The proposed approach resorts to A2C to find resource allocation strategy under different network conditions. We use the buckets-pipes game formulated in previous section as the environment interacting with the RL agent[2]. We consider episodic tasks, where an episode consists of one frame and one time slot equals to one step. The key components involved in the interactions between agent and environment are: state, reward, action, which are designed in details in the following.

1. State

We label DgNB as 0-th IAB node, and the remaining subset of IAB nodes $\mathcal{R}_{sub} = \{1, 2, \ldots, |\mathcal{R}| - 1\}$. The state at step t is a $(|\mathcal{R}| - 1)$-dimensional vector of the number of bits buffered at each IAB node n, B_n^t, normalized by the number of bits that can be transferred on a link with the minimum capacity c_{min} in a slot within the whole network, $c_{min} \cdot \delta$. This normalization is to reduce the state space so as to facilitate the exploration of RL, accelerating the convergence of the learning process. Therefore, the state at step t, $s_t = [s_n^t]_{n \in \mathcal{R}_{sub}}$, is defined as:

$$s_n^t = \left\lfloor \frac{B_n^t}{c_{min} \cdot \delta} \right\rfloor, \forall n \in \mathcal{R}_{sub}. \quad (5)$$

2. Action

The patterns generated in Section IV-A are the actions the agent can execute, which in fact contain links activation as sub-actions. In each step (slot), the agent selects a pattern according to the current policy and then the links in this pattern are activated and enabled to transmit data. If we denote the pattern set as $\mathcal{P}$, then the action at step t is $a_t \in \mathcal{P}$.

3. Reward

Our objective is to maximize the total traffic volume reaching UEs in a frame. According to this goal, the intuitive idea on designing reward function is to use total amount of bits users receive in each slot as the immediate reward. However, this strongly biases the solution toward

[1]We assume that the sizes of the queues do not limit the performance of the system.

[2]This RL agent can be implemented in a central network controller hosted at DgNB.

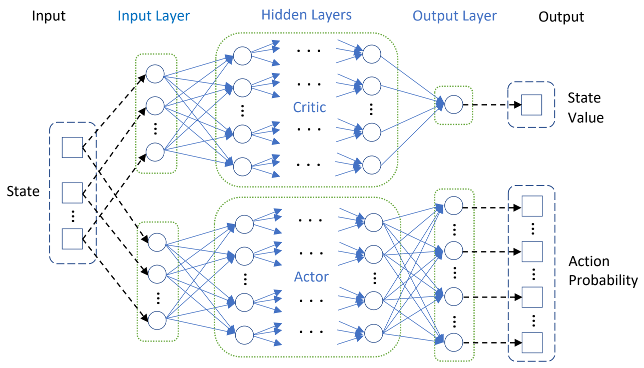


Fig. 2: A2C architecture for IAB scheduling.

UEs directly connected to the DgNB. Indeed, in IAB multi-hop network scenarios, bits received by UEs in a certain slot are the cumulative result of the bits moving through the wireless backhaul in previous slots, which does not produce any immediate data transfer. In short, the difficulties we need to overcome in defining the immediate reward involve:

- How could we precisely evaluate the current action only based on the immediate effect at the moment, instead of the accumulative effect of previous actions?
- How could we simultaneously relate throughput to reward and keep reward normalized?
- How could we avoid the bias on direct connections between DgNB and UEs?

Based on the considerations above, the reward function is designed in (6).

$$r_t = \begin{cases} \sum\limits_{\substack{\forall n \in \mathcal{R}, u \in \mathcal{U}, \\ (n,u) \in a_t}} I^t_{n,u}, & \text{if } \forall (i,j) \in a_t \text{ are effective;} \\ -\lambda, & \text{if } \exists (i,j) \in a_t \text{ is ineffective.} \end{cases} \quad (6)$$

We separate the links of the pattern selected at each slot into two categories: effective and ineffective. Effective links refer to the links which indeed transmit data. Ineffective links indicate those links, on which there are no bits really carried. This could happen when links are blocked by obstacles or there are no bits in transmitters' queues. If the entire set of links (both access and backhaul links) in the pattern selected is effective, the number of access links between RNs and UEs is used as reward; otherwise, the reward is set to a negative value $-\lambda$ as a penalty to the agent, where λ controls the extent of penalty (i.e., the bigger, the harder, by default set to 1). $I^t_{n,u}$ is a binary indicator of whether link (n,u) is effective at t: it takes value 1 if the number of bits transmitted on link (n,u) at step t is not null. This penalty approach has shown to effectively drive the agent to avoid the patterns which contain blocked links, and significantly speed up the learning process.

4. Algorithm

We apply k-step Advantage Actor Critic (A2C) [17] to our scenario rather than other RL algorithms because it can leverage merits of both value based approach and policy gradient and it empirically performs better than other similar approaches, as we found out in our preliminary tests. It is based on the state, the action taken, the reward obtained at step t (i.e., s_t, a_t and r_t). The algorithm's goal is to tune parameters ψ (which determines the value function) and θ (which determines the agent policy) in order to obtain the best actor and critic convergence. Note that vectors ψ and θ are in the form of the weights of critic and actor DNNs, respectively, which aim at approximating value and policy functions.

The critic part, whose aim is to approximate value function $v_\pi(s_t, \psi)$, is given by:

$$G^k_\pi(s_t;\theta,\psi) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k v_\pi(s_{t+k},\psi), \quad (7)$$

$$A_\pi(s_t,a_t;\theta,\psi) = G^k_\pi(s_t;\theta,\psi) - v_\pi(s_t,\psi), \quad (8)$$

$$\min_\psi A_\pi(s_t,a_t;\theta,\psi)^2, \quad (9)$$

where (7) defines the expected k-step return $G^k_\pi(s_t)$, based on k-step experience, with future reward discount factor γ, and the advantage $A_\pi(s_t,a_t)$ in (8) represents the critic loss, i.e., the difference between real reward and estimated value. Trying to minimize critic loss (9) leads value function estimation to convergence.

The actor part, since the policy's performance measure is state value (long-term reward), has the ultimate goal of guiding the policy parameter θ to the direction of increasing the state value: $\max_\theta v_{\pi_\theta}(s,\psi)$.

These two parts collaborate to converge to the best policy. The policy determines collected rewards, which in turn define the value function that drives the policy improvement.

The parameters' updates in critic and actor networks are based on the corresponding gradients. These two gradients perform updates according to (10) and (11) where $H(\pi(s_t;\theta))$ is the entropy of the actions considered by the probabilistic policy function π at state s_t and hyper-parameter β controls the importance of the entropy regularization term.

$$d\psi \leftarrow d\psi + \partial(A_\pi(s_t,a_t;\theta,\psi))^2/\partial\psi, \quad (10)$$

$$d\theta \leftarrow d\theta + \nabla_\theta \log\pi(a_i|s_i;\theta)A_\pi(s_t,a_t;\theta,\psi) + \beta\nabla_\theta H(\pi(s_t;\theta)). \quad (11)$$

The architecture of A2C is shown in Figure 2. The upper and lower parts respectively show an example of critic network and actor network. They take state vector as input. The critic network computes the input state's value, so its output layer only embraces one neuron. The actor network outputs the action probabilities which are the basis of action selection, thus its output layer has the same number of neurons as the candidate actions.

The learning procedures incorporate the flow allocation and pattern scheduling process to compute state transition and reward (see Algorithm 1). The critic and actor parameters get updated every t_{max} steps while the length of the learning period is T_{max} steps. Lines 6-16 collect data for updating gradients and parameters. A pattern (action) is chosen according to the current policy (see Line 7). The blocked links in set L_b are eliminated from the pattern chosen (Line 9), so that the remaining links can transmit an equal share of bits, limited by link capacity (Line 12) and the number of bits in the transmitter's buffer (Line 11). Reward and state transition are computed in Lines 13-15.

Lines 17-18 set the initial value of k-step return G. Then the gradients are updated in Lines 19-23. Finally, actor and critic parameters θ and ψ are updated in Line 24. The above operations are repeated until the learning phase ends, after T_{max} steps. This is an offline training procedure, which requires some computational effort but must be run only once to set the proper θ and ψ values. Then, the trained networks can be used during the IAB network operations, with much less effort, to properly drive the pattern selection.

Algorithm 1 *Learning Procedures on IAB Resource Allocation*

```
    Parameters: training steps $T_{max}$, update steps $t_{max}$.
 1: Initialize actor network $\pi(a|s,\theta)$, critic network $v(s,\psi)$;
 2: Initialize step $t \leftarrow 1$;
 3: while $t < T_{max}$ do
 4:     $t_{start} \leftarrow t$;
 5:     Get state $s_t$; Reset gradient $d\theta \leftarrow 0, d\psi \leftarrow 0$;
 6:     while $t - t_{start} < t_{max}$ and $s_t$ is not terminal do
 7:         Choose action (pattern) $a_t$ based on $\pi(a|s_t,\theta)$;
 8:         $r_t \leftarrow 0$;
 9:         Eliminate blocked link set $a_t \leftarrow a_t \setminus L_b$;
10:         for each $i \in (i,j) \in a_t$ do
11:             if $B_i^t > 0$ then
12:                 $B_j^t \leftarrow B_j^t + \min\{\frac{B_i^t}{|\{(i,\cdot)\}|}, c_{i,j} \cdot \delta\}$;
13:                 if $i \in \mathcal{R}, j \in \mathcal{U}$ then $r_t \leftarrow r_t + 1$;
14:         if $L_b \neq \emptyset$ or $\exists B_i^t = 0$ then $r_t \leftarrow -\lambda$;
15:         $s_{t+1} = [\left\lfloor \frac{B_n^t}{c_{min} \cdot \delta} \right\rfloor]_{n \in \mathcal{R}_{sub}}$;        ▷ w.r.t. (5)
16:         $t \leftarrow t + 1$;
17:     if $s_t$ is not terminal then $G \leftarrow v(s_t,\psi)$;
18:     else $G \leftarrow 0$;
19:     $i \leftarrow t - 1$;
20:     while $i \geqslant t_{start}$ do
21:         $G \leftarrow r_i + \gamma G$;
22:         Update gradients $d\theta$ and $d\psi$;   ▷ w.r.t. (10),(11)
23:         $i \leftarrow i - 1$;
24:     Update $\theta$ and $\psi$ using $d\theta$ and $d\psi$;
```

5. Online Model Framework

Here we present an online learning scheme based on Algorithm 1. In offline case, a model is pre-trained offline and hereafter applied to a network scenario, which meets challenges if the network status drastically changes. Note that here we do not refer to the mere link availability change, which, provided its blockage probability p_b does not significantly change, can be addressed by an offline training. We rather refer to a more dramatic change in which the entire distribution of link blockage probability does change.

One feasible but time-consuming solution is to train a model again from the very beginning. However, a more attractive choice is to conduct training and application of a model in parallel. The data collected from the application is directly used in the training course, while at the meantime, the trained model so far is applied to the network scenario. In this way, the model can catch system dynamics on-the-fly and adjust the training part to adapt to the new environment. We believe that the online model update can be implemented on a time slot basis with optimized coding and specialized hardware, like GPUs that would allow to set the per-slot computation time much below the 1 ms required by our general-purpose system. This is even more realistic considering the limit of 10 IAB nodes proposed by the standardization. Nevertheless, our approach can be adapted by considering the time granularity of a small set of slots, lasting enough to complete the update.

Experimental results in Section V demonstrate that this framework makes the model more robust to the dynamic environments. Online trained model can work seamlessly for low-impact changes and recover fast (w.r.t. the physical network timing) from the drop caused by strong changes.

V. Experimental Results

In this section, the proposed A2C based approach on IAB flow allocation and pattern scheduling is evaluated. We first introduce testing scenarios and model settings. Then we evaluate our approach by applying the offline algorithm to scenarios with and without blockages and comparing with random pattern selection method (RANDOM) and optimization CG-based method (OPT) described in Section IV-A in terms of achieved throughput. Finally, we test the online algorithm which is able to adapt to dynamic IAB networks where different number of links exhibit random blockages. In particular, we compare the offline and online model, demonstrating that the online model can automatically recover from sudden changes, outperforming the solution based on offline algorithm. The models are trained on a machine with an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, RAM 126GB.

A. IAB Network Scenario

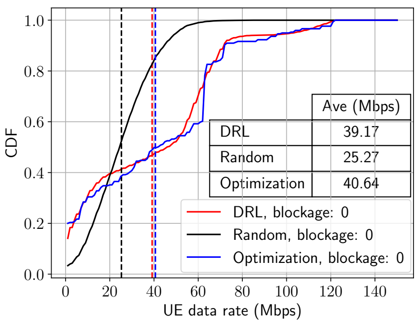
The tests presented are based on random scenarios produced by an instance generator for FWA playground provided by a worldwide mmWave device manufacturer. The generator is compliant with 3GPP NR IAB simulation guidelines [1]. We consider 300×300 square cell with 1 DgNB, 4 and 30 randomly deployed IAB nodes and UEs. The 3GPP NR TR 38.901 channel model [18] is adopted. In-band IAB is at 28 GHz, 400 MHz of bandwidth, and NR Numerology #3 (120 kHz subcarrier spacing). Each frame embraces 80 slots, each of which lasts 125 μs. We consider fixed MCSs as 16 and 8 respectively for backhaul and access. MCS 16 corresponds to SINR threshold 5.60 dB and capacity 525.9 Mbps, while MCS 8 can achieve capacity 121.4 Mbps with SINR threshold -3.77 dB.

B. Model Settings

RMSOptimizer is used in DNN training. Offline and online models use different DNN settings including neurons N, layers L, learning rate lr, batch size bs, and RL settings like discount factor γ . Respectively, $32, 8, 0.002, 200, 0.99$ are for offline and $32, 4, 0.0007, 500, 0.99$ are for online. Note that critic and actor networks use the same DNN settings in each model. The output layers in critic and actor networks employ respectively linear and softmax functions.

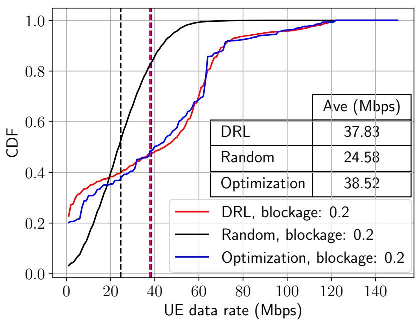
C. Performance Analysis on Offline Model

In these tests, the proposed A2C-based model (DRL) is trained offline and then applied to specific network scenarios. We first analyze the case with all the links available. And



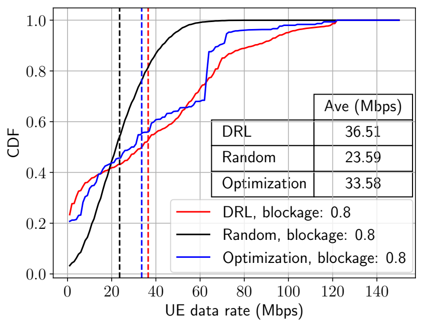
	Ave (Mbps)
DRL	39.17
Random	25.27
Optimization	40.64

(a) UE rate CDF (no blockage).



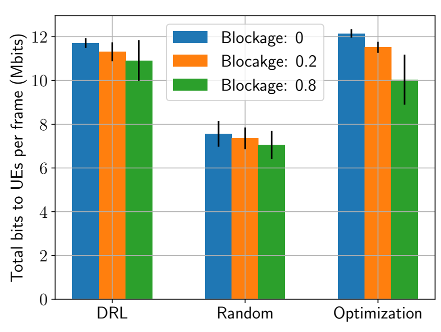
	Ave (Mbps)
DRL	37.83
Random	24.58
Optimization	38.52

(b) UE rate CDF (blockage: 0.2).

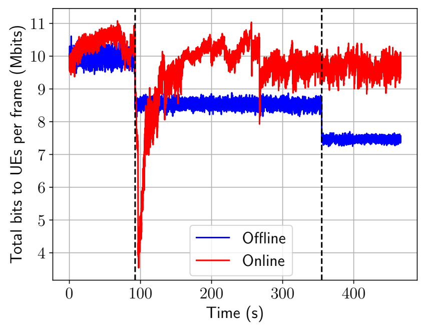


	Ave (Mbps)
DRL	36.51
Random	23.59
Optimization	33.58

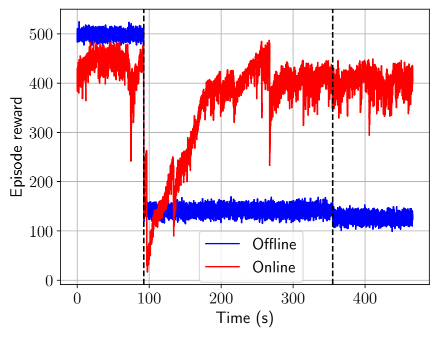
(c) UE rate CDF (blockage: 0.8).



(d) Total number of transferred bits.



(e) Total number of bits in real time.



(f) Episode reward in real time.

Fig. 3: Performance of offline and online models. (a)-(d) show the comparison between offline DRL, RANDOM and OPT. (e)-(f) focus on DRL and compare its online and offline versions in terms of transferred volumes and episode reward.

then we randomly block several links (e.g., 4 links out of around 80 links) with different probability (i.e., 0.2 and 0.8), hence these links behave up and down. We employ random pattern selection strategy (RANDOM) and optimization CG-based method (OPT) as benchmarks. OPT provides the near-optimal benchmark when dealing with static and ideal networks. The tests are performed on 10 randomly generated instances.

1. UE Data Rate CDF

The cumulative distribution functions (CDFs) over UEs' achievable data rates are shown in Figures 3(a)-3(c). The vertical dashed lines indicate the average UE data rates that are also reported in the tables attached to the figures.

As we can see, under both non-blocked and blocked conditions, the maximum rates achieved by DRL and OPT are near 121.4 Mbps, the maximum rate achievable with MCS 8. Whereas, RANDOM obtains at most around 60 Mbps. Comparing OPT and DRL CDFs, we can see that maximum data rates do not significantly differ, which is reasonable because a few link blockages are less likely to prevent the other links to provide maximum rates to UEs. In terms of average UE rates, DRL can approximate OPT, while RANDOM performs much worse. As we randomly pick 4 links to impose blockages with probability 0.2, the gap between DRL and OPT is reduced. Due to the mild blockages, OPT still has a small lead, as the actual scenario is still very similar to the one assumed in the optimization, i.e., all links are always available. However, after the probability is raised to 0.8, DRL outperforms OPT by about 10%, even if the blockages involve only 4 links over the entire set of about 80 links.

2. Traffic Volumes

A more direct way to evaluate models' quality is the traffic volume transferred from DgNB to UEs in a frame. Figure 3(d) shows clear differences between the three methods and their behaviors in the three blockage cases. 1) For all three methods, the total amount of bits delivered to UEs decreases as more blockages are introduced into the network. The descending order of the throughput reduction degree for the three methods is OPT $>$ DRL $>$ RANDOM. This is because the method with better performance is more likely to be scenario-specific. 2) DRL transfers more bits than OPT after the blockage probability is raised to 0.8. This is due to the adaptability of the DRL to scenarios with blockages, considering system dynamics. The RL agent can learn the availability level of each link during the training phase and tune the pattern selection strategy accordingly. 3) The standard deviations marked by black vertical lines on the top of bars tell us the range of values assumed by the solutions of different network instances. With blockage probabilities varying in $\{0, 0.2, 0.8\}$, apart from RANDOM whose oscillation is constant, DRL and OPT display increasing oscillations as the blockage probability increases. This is due to the cases in which blockages involve critical links, causing a larger performance decrease.

D. Performance Analysis on Online Model

In this final experimental analysis, we test the online training framework where a model is simultaneously trained and applied to the IAB network. We intend to assess whether the model can on-the-fly adapt to the dynamic networks with consecutive changes in the link blockage probability distribution. We show here an example on a single instance used in offline experiments, however similar conclusions can be made for the other instances. Our goal is to understand whether the online model can keep stable and possibly re-adapt when more and more network links are exposed to interruption.

We plot episode rewards and traffic volumes to, respectively, evaluate the model learning process and its performance in practice. Figures 3(f)-3(e) show the variations of these two metrics as time goes on. Vertical dashed lines indicate the moments when we change the blockage situation. Here we keep the same blockage probability 0.8 in the whole process and block 4 links at the beginning, $t = 0$. At this starting point, both approaches have reached a stable working point: the offline model has been pre-trained using this initial environment, while the online model has converged to a stable condition. At the first vertical line, we introduce blockages into another 4 heavily-used links, so till now 8 links are blocked. At the second vertical line, we impose blockages on further 4 links that are not very often used, hence totally 12 links are blocked.

We can notice that the first change causes to both online model and pre-trained offline model a big performance drop. However, online model can quickly recover while offline model's performance remains low. After the second change, only the offline model sees a decrease, while the online model proceeds with actual no variation. These evidences prove that the online model could effectively capture system dynamics and make decisions based on a long-term merit, while the offline model loses its learning capability.

VI. Conclusion

In this paper, we have proposed an RL-based resource allocation algorithm which is able to cope with the dynamics of the link status in mmWave 5G IAB networks. The results have shown that this approach can outperform the traditional optimization approaches typically used in wireless multi-hop networks. Indeed, our algorithm can automatically adapt to environmental changes and rapidly recover from blockage situations. We are currently investigating some extensions of our approach to consider alternative channel blockage models and potential correlations among link failures.

Acknowledgment

This work was partially supported by China Scholarship Council (CSC) Grant No. 201806470077.

References

[1] "Study on integrated access and backhaul," 3GPP TR 38.874, Tech. Rep.

[2] D. Yuan, H.-Y. Lin, J. Widmer, and M. Hollick, "Optimal joint routing and scheduling in millimeter-wave cellular networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 1205–1213.

[3] A. Capone, G. Carello, I. Filippini, S. Gualandi, and F. Malucelli, "Routing, scheduling and channel assignment in wireless mesh networks: Optimization models and algorithms," *Ad Hoc Networks*, vol. 8, no. 6, pp. 545–563, 2010.

[4] A. Capone, I. Filippini, S. Gualandi, and D. Yuan, "Resource optimization in multi-radio multi-channel wireless mesh networks," in, Wiley Online Library, 2013.

[5] M. Polese, M. Giordani, A. Roy, D. Castor, and M. Zorzi, "Distributed path selection strategies for integrated access and backhaul at mmwaves," in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–7.

[6] T. K. Vu, C.-F. Liu, M. Bennis, M. Debbah, and M. Latva-Aho, "Path selection and rate allocation in self-backhauled mmwave networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2018, pp. 1–6.

[7] J. Du, E. Onaran, D. Chizhik, S. Venkatesan, and R. A. Valenzuela, "Gbps user rates using mmwave relayed backhaul with high-gain antennas," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1363–1372, 2017.

[8] Q. Hu and D. M. Blough, "Relay selection and scheduling for millimeter wave backhaul in urban environments," in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, IEEE, 2017, pp. 206–214.

[9] M. Feng and S. Mao, "Dealing with limited backhaul capacity in millimeter-wave systems: A deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 50–55, 2019.

[10] Y. Niu, C. Gao, Y. Li, L. Su, D. Jin, Y. Zhu, and D. O. Wu, "Energy-efficient scheduling for mmwave backhauling of small cells in heterogeneous cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 2674–2687, 2016.

[11] Y. Li, J. Luo, W. Xu, N. Vucic, E. Pateromichelakis, and G. Caire, "A joint scheduling and resource allocation scheme for millimeter wave heterogeneous networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2017, pp. 1–6.

[12] J. Garcıa-Rois, R. Banirazi, F. J. González-Castaño, B. Lorenzo, and J. C. Burguillo, "Delay-aware optimization framework for proportional flow delay differentiation in millimeter-wave backhaul cellular networks," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 2037–2051, 2018.

[13] A. Ortiz, A. Asadi, G. H. Sim, D. Steinmetzer, and M. Hollick, "Scaros: A scalable and robust self-backhauling solution for highly dynamic millimeter-wave networks," *IEEE Journal on Selected Areas in Communications*, 2019.

[14] T. K. Vu, M. Bennis, M. Debbah, M. Latva-Aho, and C. S. Hong, "Ultra-reliable communication in 5G mmwave networks: A risk-sensitive approach," *IEEE Communications Letters*, vol. 22, no. 4, pp. 708–711, 2018.

[15] B. Sahoo, C.-H. Yao, and H.-Y. Wei, "Millimeter-wave multi-hop wireless backhauling for 5G cellular networks," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, IEEE, 2017, pp. 1–5.

[16] M. Saad and S. Abdallah, "On millimeter wave 5G backhaul link scheduling," *IEEE Access*, vol. 7, pp. 76 448–76 457, 2019.

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[18] "Study on channel model for frequencies from 0.5 to 100 ghz," 3GPP TR 38.901, Tech. Rep.