PHaul: A PPO-based forwarding agent for Sub6 enhanced Integrated Acess and Backhaul networks

Jorge Pueyo
I2CAT Foundation
jorge.pueyo@i2cat.net

Daniel Camps-Mur
I2CAT Foundation
daniel.camps@i2cat.net

Miguel Catalan-Cid
I2CAT Foundation
miguel.catalan@i2cat.net

***Abstract*—3GPP Integrated Access and Backhaul (IAB) allows operators to deploy outdoor mm-wave access networks in a cost-efficient manner, by reusing the same spectrum in access and backhaul. In IAB networks the performance bottleneck is the wireless backhaul segment, where efficient forwarding strategies are needed to effectively use the available capacity. In addition, the performance of the mm-wave IAB backhaul segment is contingent on the availability of line of sight (LoS) conditions in the selected deployment sites. To mitigate LoS dependence, in this paper, we propose to complement the mm-wave backhaul segment of IAB networks with additional Sub6 backhaul links, which contribute to the capacity and robustness of the backhaul network. We refer to IAB networks combining Sub6 and mm-wave links in the backhaul as Sub6 enhanced IAB networks. In this context, the main contribution of this paper is PHaul, a forwarding engine for Sub6 enhanced IAB networks that accomodates different traffic engineering criteria, and combines an offline path selection heuristic with an online Deep Reinforcement Learning (DRL) agent based on Proximal Policy Optimization (PPO). By leveraging a network digital twin of the IAB wireless backhaul, PHaul periodically samples the input traffic of the backhaul network and updates flow to path mappings, with execution times below 10 seconds in realistic backhaul topologies. We present an exhaustive performance evaluation, where we demonstrate that PHaul can achieve gains of up to 36% in throughput efficiency and of up to 20% in fairness, when compared against two alternative heuristics in a wide range of network configurations. We also demonstrate that PHaul is robust to differences between the network topologies considered in the training and inference phases, which can occur in practice due to link failures.**

***Index Terms*—3GPP IAB, Sub6 and mm-wave, Deep Reinforcement Learning (DRL), Proximal Policy Optimization (PPO)**

I. Introduction

Addressing the data rate demands of future communication services will require heterogeneous 5G-Advanced (5G-A) networks with a dense high-capacity layer composed of outdoor mm-wave (26/28 GHz) small cells [1]. This trend is expected to continue in 6G networks, where the Sub-THz band (90GHz -300GHz) is being considered to deliver even greater access capacities [2].

Laying out dedicated fibre for backhauling is a roadblock to enabling a massive deployment of outdoor small cells. To address this challenge 3GPP introduced the Integrated Access and Backhaul (IAB) technology in Release 16 [3]. The idea of IAB is to enable an operator to start by deploying a small number of fibre connected mm-wave small cells, known as *donor* nodes in IAB terminology, and to extend service coverage by deploying additional *IAB nodes* that are wirelessly backhauled through donor nodes, and can provide direct service to User Equipments (UEs), or act as parent nodes for other IAB nodes. When service demand grows, an operator can add fibre connectivity to IAB nodes, upgrading them to donor nodes, thus increasing the effective backhaul capacity in a cost-efficient manner.

While the IAB model defined by 3GPP is spectrum agnostic and can operate either in the above-6 GHz or sub-6 GHz spectrum, the main use case for IAB networks is to provide mm-wave spectrum in the access and backhaul, complementing the Sub6 spectrum offered by the macro-cell layer. However, mm-wave backhauling in urban environments is often subject to link blocking issues, and to non-line of sight performance degradation [4]. Therefore, in this work we consider combining Sub6 spectrum together with mm-wave frequencies in the IAB backhaul segment to address both coverage and capacity requirements in dense urban deployments of small cells. Limiting the use of Sub6 in IAB to the backhaul segment reduces potential cross-interference with the macro-layer, because backhaul links are fixed and can be engineered to avoid interference. In addition, the upper part of the 6 GHz band, which was recently authorized for IMT use in the WRC23 conference [5], could be exploited for the IAB backhaul, avoiding any interference with the macro layer. We refer to IAB networks combining Sub6 and mm-wave links in the backhaul as Sub6 enhanced IAB networks.

When planning a traditional wireless backhaul network, a common approach is to overprovision based on busy hour. However, this is not a good approach for IAB networks in urban environments, for the following reasons. First, the limited coverage of the mm-wave access results in quick variations in the number of UEs connected to a given cell, which complicates the estimation of the required backhaul capacity. Second, overprovisioning the IAB backhaul is not economically efficient as it removes away access capacity, thus requiring the deployment of additional IAB nodes. Hence, if overprovisioning is not a practical approach for planning IAB networks, it is critical to consider flexible routing schemes that can maximize the efficiency of the wireless backhaul by

This work was funded by the European Commission through the SNS JU project NANCY (grant agreement No. 101096456). The authors acknowledge CERCA Programme/Generalitat de Catalunya for sponsoring part of this work.

adapting backhaul paths to the available load demands at a given time.

To route packets in IAB-based networks, 3GPP has defined the *Backhaul Adaptation Protocol* (BAP) that is based on source routing. Namely, a set of paths between each IAB node and its target donor node are pre-computed and the next hop entry for each path is programmed into the forwarding engine of each IAB node. Thus, higher layer packets arriving at an IAB node are matched to their corresponding path according to their intended destination. This source-based routing model can be easily extended to a heterogeneous backhaul network that comprises both mm-wave and Sub6 frequencies, including within the set of available paths between an IAB node and a donor, both Sub6 and mm-wave paths. BAP can be used both in tree and mesh topologies, where mesh topologies can be supported in IAB by having an IAB node connect to more than one IAB parent node, for example by including multiple IAB Mobile Termination (MT) functions in the same node, or by means of using Dual Connectivity (DC) [3].

Under the BAP forwarding model, even though paths are precomputed, the classifier matching higher layer flows into the pre-provisioned backhaul paths can be dynamically updated. The main focus of this paper is to investigate mechanisms that dynamically update the matching between backhaul flows and available paths in Sub6 enhanced IAB backhaul networks, to optimize a given traffic engineering criteria. In particular, we investigate the use of Deep Reinforcement Learning (DRL) to design a forwarding agent that, once trained on a given IAB backhaul topology, is able to dynamically update the flow to path mappings according to varying traffic demands.

The main contribution of this paper is threefold.

i. First, we present a novel IAB architecture that combines Sub6 and mm-wave links in the backhaul segment.
ii. Second, we present PHaul, an IAB forwarding engine that combines an offline path selection heuristic with an online DRL agent based on Proximal Policy Optimization (PPO) [6]. PHaul supports flexible traffic engineering criteria and leverages a network digital twin to perform path allocations in Sub6 enhanced IAB networks.
iii. We describe the training process of PHaul, and we perform an exhaustive simulation-based evaluation comparing PHaul to two alternative greedy algorithms, both in terms of performance and inference time.
iv. Finally, to ease reproducibility, we have open sourced our implementation of PHaul[1], as well as the environments we have created to evaluate its performance.

The paper is structured as follows. Section II describes related work. Section III presents the considered network model, Section IV describes the design of PHaul, and Section V evaluates the performance of PHaul against potential alternatives. Finally, Section VI summarizes and concludes the paper.

[1]https://github.com/Fundacio-i2CAT/phaul/

II. Related Work

We survey the state of the art relevant to this work grouped in: i) works that explore the utilization and combination of Sub6 and mm-wave bands for wireless backhauling, ii) works that explore the use of Machine Learning (ML) mechanisms for routing, and iii) works that focus on flow routing in IAB networks.

Although usually focused on mm-wave bands and fixed IAB nodes, several works propose novel architectures to enhance IAB performance in specific scenarios. Authors in [7] present and evaluate through simulations an IAB architecture using Sub6 bands to support wide coverage and mobile systems. In [8] the authors consider Sub6 IAB to introduce a novel solution to allocate access and backhaul resources. The solution combines centralized (IAB donor) and distributed (IAB nodes) approaches to handle both non-bursty and bursty traffic, respectively. Authors in [9] introduce a dynamic resource management framework for UAV-assisted IAB networks utilizing Reconfigurable Intelligent Surfaces (RIS). This framework uses a distributed Stackelberg game-theoretic approach to optimize end-to-end energy efficiency by exploiting the integration of UAVs with IAB and RIS technologies.

Regarding the combination of Sub6 and mm-wave bands in the wireless backhaul, the authors in [10] demonstrate a city-wide deployment that combines IEEE 802.11ad radios in the 60GHz band, with IEEE 802.11 radios in the 5 GHz band. This approach however is not based on IAB. Instead, an SDN-based control plane is used to implement source-based routing. In [11] the authors present a centralized SDN-based wireless backhaul system, also based on IEEE 802.11 radios in the 5 GHz and 60 GHz bands, which allocates for each flow a main and a backup path using heuristic policies that search across a set of K predefined paths between each source-destination pair. The authors do not report on the execution times of their proposed heuristic.

The application of ML-based techniques for routing is motivated by the large computational costs of applying optimal routing strategies. An optimal solution to the path allocation problem with traffic engineering constraints, given a known topology and demand matrix, is described in [12] through a Multi-Commodity Flow Problem (MCFP) formulation where flows in each link share bandwidth using a max-min fairness criteria. This method is shown to take as long as one hour for some problem instances, thus not being practical in wireless backhaul networks with constantly changing demands. To achieve lower execution times, recent works in the state of the art have started to study the application of Machine Learning (ML) techniques. Most of these works adopt a Reinforcement Learning (RL) framework, where a routing agent is considered that selects a path or next-hop as action, models the environment using counters available in routers or switches, and defines a reward that captures a particular routing objective.

In [13], R2L, a centralized routing agent, is presented that takes a next-hop decision on each new packet arrival at a

TABLE I
SUMMARY OF THE STATE OF THE ART

Reference Number	Main contribution	Routing/Scheduling solution	Performance evaluation
[8]	IAB network (only sub6), avoids conflict in resource allocation for bursty-traffic	Centralized donor allocates hard resources (non-bursty traffic), IAB nodes apply distributed allocation (bursty- traffic)	Simulation: GBR-only and bursty traffic scenarios
[9]	Integrates RIS and UAV-assisted IAB, targets energy efficiency	Leader UAV determines RIS configuration, bandwidth splitting, UAV Tx power and initial UE Tx Power. UEs determine optimal power according to bandwidth. Stackelberg game-based iteration.	Modeling: Convergence of the optimisation process. Simulation: single UAV, RIS and BS, comparison of energy-efficiency vs data rate approaches
[11]	SDN-based Wi-Fi backhaul, centralized per-flow path computation plus distributed fast-recovery when links fail	Computes a subset of paths per flow, selects main and backup path according to network status and flow requirements, creates forwarding rules to apply fast local link reroute	Modeling: analysis of number of rules and signaling overhead. Simulation: performance evaluation according to network size. Experimental: Link Failure detection time, E2E evaluation in a real testbed
[13]	Wired network, per packet decision on next hop, centralized RL agent	RL based on neural network and evolutionary algorithm. State: local network measurements (e.g. queue size) and packets header. Action: outgoing port. Reward: configurable (e.g. delay or throughput)	Simulation: fixed number of nodes (5 or 16), comparison with shortest path, longest path and load balancing approaches
[14]	Wired network, per packet decision on next hop, one RL agent per router	RL based on Q-learning. State: N-Optimal paths, topology, link state, delay measurements. Action: path selection. Reward: latency	Simulation: Comparison with other approaches in terms of scalability, robustness and delay.
[15]	Industrial IoT networks, joins routing and TDMA scheduling, centralized RL agent	RL based on PPO. State: State of transmissions in each node at a given slot (queue length, time remaining, hops remaining, priority), precomputed paths. Action: Chooses transmissions in the next slot (according to 6 criteria) and their next hop. Reward: delay value weighted by deadline delay and delivery status	Simulations: impact of number of routes, channels and nodes. Performance in terms of packet delivery ratio, impact of untrained topologies
[16]	IAB network (only mmWave), joins routing and resource allocation, one RL agent per IAB node	RL based on Soft Actor-Critic algorithm. State: information from neighbor nodes (channel condition, latency, reliability). Action: next hop in DL and UL. Reward: Combination of latency and reliability.	Simulations: 1 donor, 18 IAB nodes, variable number of UEs. Computational complexity. Performance in terms of average delay and reliability
Phaul	IAB network (mmwave and sub6), per-flow path allocation, centralized RL agent	RL based on PPO. State: Precomputed N-optimal paths (least common heuristic) and input data rate per flow, current path and measured effective data rate (digital twin). Action: Path selection. Rewards: Configurable according to throughput efficiency and fairness	Simulations: 3 donors, up to 60 IAB nodes, variable topologies and offered load. Impact of path computation heuristics, training steps, broken links and untrained topologies. Performance in terms of throughput efficiency and fairness.

wired router. The agent can be trained to optimise different criteria (e.g. throughput, delay), and network state is built from the queue lengths at different routers. The agent is shown to converge in the training phase, although only a simplified network topology is considered in the evaluation. In [14] a traditional Q-learning algorithm is used to select among a set of N paths between each source and destination in a wide area network. The agent is trained with the objective of choosing the path with lowest delay. Closest to our work is the RECCE agent proposed in [15], which however focuses on a different use case, namely industrial WirelessHart IoT networks. RECCE is based on a modern Proximal Policy Optimization (PPO) agent and performs joint routing and scheduling by choosing among a set of γ precomputed paths and among six different scheduling strategies. A single reward is proposed to maximize the number of packets delivered within their deadline.

Looking at routing for IAB networks the authors in [16] propose a DRL scheme for both routing and resource allocation. In their proposal each IAB node embeds an independent DRL agent that selects the next next hop in uplink and downlink. Unlike this proposal, in this paper we propose a global DRL agent that performs routing decisions considering the network as a whole. In [17] the authors propose a genetic algorithm to assit the planning stage of IAB networks, and a routing heuristic to cope with temporary link blockages. This paper though does not address the problem of load balancing in IAB networks, which is the main focus of our work.

To the best of our knowledge, our work is the first to address the problem of ML-based routing in Sub6 enhanced IAB networks. Table I summarizes the main features of the solutions presented in this section, providing a comparison with our Phaul proposal.

III. Network Model

Figure 1 depicts the network model considered in this work, consisting of an IAB network where the backhaul segment is enhanced with additional Sub6 backhaul links. In the bottom left of Figure 1 we depict the logical architecture of a Sub6-enhanced IAB node, which, following the IAB architecture, features a Distributed Unit (DU) and Mobile Terminated (MT) function both for the Sub6 and the mm-wave bands. However, only the mm-wave DU will be used to connect UEs in the access, as Sub6 access coverage is already provided by the macro layer.

The central part of Figure 1 depicts the following elements:

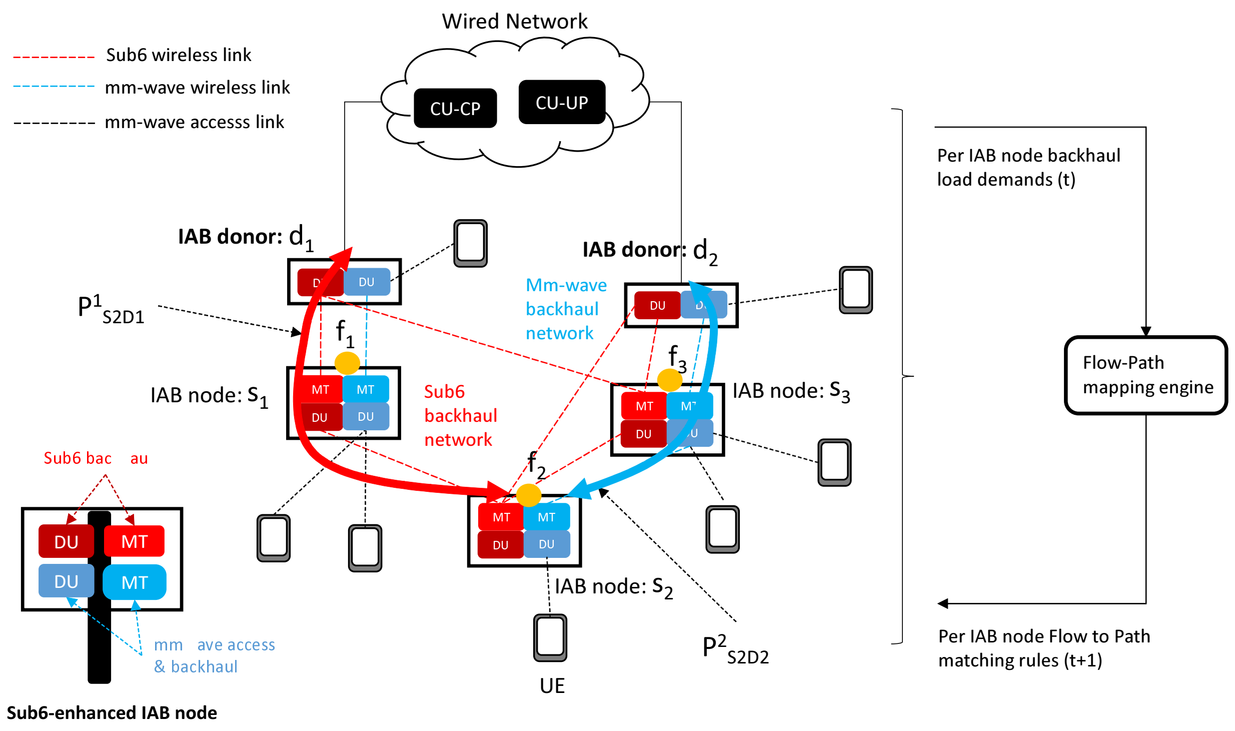


Fig. 1. PHaul network model

i. IAB donor nodes, denoted as $d_k \in \mathbb{D}$, featuring both a Sub6 and a mm-wave DUs, which connect UEs and other IAB nodes to the wired network, where the Centralized Unit (CU) components are located.
ii. IAB nodes, denoted as $s_i \in \mathbb{S}$, which provide access to UEs and other IAB nodes. An IAB node may directly connect to an IAB donor or to another IAB node. Multi-path is supported by embedding multiple MT functions into a single IAB node or by using Dual Connectivity, and allows an IAB node to establish more than one link to other IAB donor or IAB parent nodes. IAB nodes provide access to multiple UEs, where each UE establishes a separate PDU session. However, we consider that in the backhaul, traffic from all UEs is aggregated into a single backhaul PDU session, which we refer to as a *backhaul flow* $f_i \in \mathbb{F}$.
iii. Backhaul flows, $f_i \in \mathbb{F}$, originate at non-donor IAB nodes, i.e. $s_i \in \mathbb{S}$. Each backhaul flow carries a time varying traffic load, $\lambda_i(t)$, generated by all the UEs in RRC_CONNECTED state. Each backhaul flow is mapped to a pre-provisioned IAB *backhaul path* defined between an IAB node $s_i \in \mathbb{S}$ and an IAB donor $d_i \in \mathbb{D}$. For each flow f_i, the source is fixed, i.e. s_i, however any donor d_k could be used as destination, as long as a backhaul path exists. The reason is that all donor nodes provide access to the wired network where the CU resides.
iv. The Sub6 backhaul, shown in red in Figure 1, and the mm-wave backhaul, shown in blue in Figure 1, provide a set of backhaul paths between IAB nodes and IAB donors. Due to different propagation characteristics in the Sub6 and mm-wave bands, different paths may be available in the Sub6 and mm-wave backhaul networks. The BAP protocol in IAB uses source-based forwarding. Hence, each flow $f_i \in \mathbb{F}$ is bound to use a single path, where paths are pre-provisioned. In particular, we consider that a set of $k \leq K^{max}$ paths are pre-provisioned for each flow in the Sub6 and the mm-wave backhaul networks, where $P^n_{s_i,d_k}$ indicates the n-th pre-provisioned path for flow f_i between s_i and d_k. Thus, a given backhaul flow could be routed across a total of $2K^{max}$ paths considering both networks, but can only use a single path at a given time.
v. Finally, a flow-path mapping engine is defined as a control plane entity that periodically obtains the state of the network, e.g., through reading load counters in the IAB nodes. Based on the obtained information, the flow-path mapping engine updates the mappings between backhaul flows and backhaul paths in each IAB node. The PHaul agent resides in the flow-path mapping engine.

To model the capacity available in the wireless backhaul, we consider a fixed link capacity and neglect cross-link interference. This assumption is possible in the case of IAB mm-wave links by allocating dedicated slots to the backhaul links and employing beamforming to avoid cross-link interference [3]. In the case of the Sub6 backhaul links, we consider the use of the 6GHz band, where 29 channels of 40 MHz are available [18], without legacy Wi-Fi devices, which can be used to avoid cross-link interference in practical urban wireless backhaul networks. The exact capacities of the Sub6 and mm-wave links depend on the specific radio configuration, which will be described in detail in Section V.

Notice that various backhaul flows can traverse the same backhaul node and compete for the capacity of a given Sub6 or mm-wave backhaul link. In this case, we assume that backhaul nodes assign capacity to each flow traversing a backhaul link using a max-min fairness criteria [19], which is a good approximation when competing flows through a bottleneck carry TCP traffic [20].

Based on the described network model we can define the set of paths available to flow $f_i \in \mathbb{F}$ as $\mathbb{P}_i^{selected}$, with $|\mathbb{P}_i^{selected}| \leq 2K^{max}$. Periodically, the flow-path mapping engine samples the traffic matrix from each backhaul flow and updates the per-flow path allocations. The goal of PHaul is to, based on the current traffic matrix $\{\lambda_i(t)\}$, assign for each flow $f_i \in \mathbb{F}$ a single path $P_{s_i,d_i}^{opt} \in \mathbb{P}_i^{selected}$, to optimize a given traffic engineering criteria J^{TE}. The execution time of the PHaul agent is the key factor to determine how often path allocations can be adapted to the varying traffic matrices.

Regarding the traffic engineering criteria, we consider J^{TE} to be a varying objective function defined by the operator of the IAB network as a function of the effective data rates, $\phi_i(t)$, allocated to each backhaul flow, where the effective data rate represents the actual rate that can be served from that flow, as compared to the overall flow demand represented by $\lambda_i(t)$.

A key requirement in the design of PHaul is to be able to operate with any traffic engineering criteria defined by the network operator. For example, in this work we consider the following J^{TE} criteria:

i. *Throughput efficiency*:

$$0 \leq J^{TE} = \frac{\sum_i \phi_i(t)}{\sum_i \lambda_i(t)} \leq 1 \tag{1}$$

where the goal is to maximize the proportion of load that can be carried by the network. This traffic engineering criteria would be appropriate in networks where demand is expected to be heterogeneous across IAB nodes. For example, a city center where some IAB nodes are covering a tourist hotspot, whereas other nodes cover more quiet areas.

ii. *Fairness*:

$$0 \leq J^{TE} = \frac{(\sum_i \phi_i(t))^2}{|\mathbb{F}| \sum_i \phi_i(t)^2} \leq 1 \tag{2}$$

where $|\mathbb{F}|$ is the number of flows, and the goal is to maximize fairness across flows. This traffic engineering criteria would be appropriate when demand across IAB nodes is expected to be uniform, for example when an IAB network is used to cover a suburban area.

iii. *Weighted*:

$$0 \leq J^{TE} = (1+\gamma)\frac{\sum_i \phi_i(t)}{\sum_i \lambda_i(t)} + (1-\gamma)\frac{(\sum_i \phi_i(t))^2}{|\mathbb{F}| \sum_i \phi_i(t)^2} \leq 2 \tag{3}$$

where the the parameter $-1 < \gamma \leq 1$ allows the operator to balance throughput and fairness according to its business goals. Being the more general option, we use this traffic engineering criteria with different values of γ in the rest of the paper.

IV. PHaul design

A. Design Principles

As introduced in Section III, the goal of PHaul is to perform per-flow path allocations that optimize a traffic engineering criteria, J^{TE}, with execution times in the order of few seconds to track variations in the IAB access traffic. As described in Section II, traditional optimization methods cannot be applied to this problem as they lead to excessive computational times. To address this problem, we turn our attention to Deep Reinforcement Learning (DRL).

The goal of the PHaul DRL agent is to periodically collect traffic demands for all flows, i.e. $\{\lambda_i(t)\}$, and make a path allocation decision, i.e. select $P_{s_i,d_i}^{opt} \in \mathbb{P}_i^{selected}$, $\forall f_i \in |\mathbb{F}|$. Given the network model introduced in Section III, a naive DRL implementation that jointly allocates all flows in each action, results in an action space of size $|\mathbb{A}| = (2K^{max})^{|\mathbb{F}|}$, where $|\mathbb{F}|$ is the number of flows, and K^{max} the number of available paths per flow in the Sub6 and mm-wave networks. This approach is not feasible for practical networks, as the action space can quickly become too large.

The main insight of PHaul is an approach to reduce the size of the action space, while still being able to allocate a path for each flow. The idea is to define a *digital twin* of the IAB backhaul network over which we can define a reduced action space, which consists of the allocation of a path for a single flow, instead of a joint allocation for the $|\mathbb{F}|$ flows. The agent then observes the resulting reward of the applied action over the digital twin, and continues sampling the reduced action space until an allocation for all the flows is obtained, which can then be programmed over the real network. The rationale behind this design approach is the following:

i. The topology and link characteristics of the Sub6 enhanced IAB network are expected to be stable, and can therefore be easily reproduced by a digital twin (e.g. a network simulator). If the topology changes significantly, e.g. due to link failure, then the digital twin can be correspondingly updated.
ii. The traffic demands $\{\lambda_i(t)\}$, can also be considered stable for periods of several seconds and can therefore be modelled in the digital twin. This is the time budget available to perform an allocation decision.
iii. Reducing the action space, i.e. allocating a path for only one flow at each action, simplifies training and leads to better convergence properties of the DRL agent.
iv. The repeated application of the simplified agent over the digital twin should allow to sample the larger action space leading to well performing allocations.

In Section V we validate empirically whether the previous assumptions lead to a well performing forwarding agent.

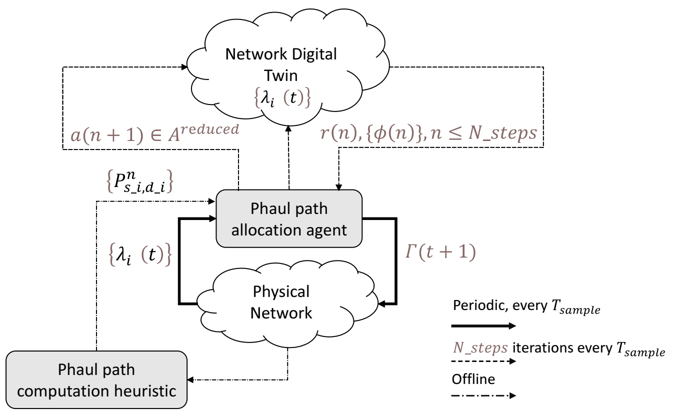

Network Digital Twin
$\{\lambda_i\ (t)\}$
$a(n+1) \in A^{reduced}$
$r(n), \{\phi(n)\}, n \leq N\_steps$
$\{P^n_{s\_i,d\_i}\}$
Phaul path allocation agent
$\{\lambda_i\ (t)\}$
$\Gamma(t+1)$
Physical Network
Periodic, every T_{sample}
N_steps iterations every T_{sample}
Offline
Phaul path computation heuristic

Fig. 2. PHaul agent design

Figure 2 describes the high level operation of PHaul. First, we distinguish the two modules that compose PHaul, namely: i) the *path computation heuristic*, and ii) the *path allocation agent*. The path computation heuristic operates offline, and its goal is to examine the topology of the physical network to derive a set of up to K^{max} paths for each backhaul flow through both the Sub6 and the mm-wave backhaul networks. We refer to this set of potential paths as $\mathbb{P}_i^{selected}$. The obtained set of paths is only recomputed when the topology of the physical network changes.

The path allocation agent operates online and maintains two loops, one with the physical network and another one with the network digital twin. The agent samples the physical network every T_{sample} to obtain a traffic matrix $\{\lambda_i(t)\}$. Subsequently, for each new sampled traffic matrix, the path allocation engine initializes the digital twin of the wireless backhaul network, and performs up to N^{steps} interactions with the network digital twin to obtain the final flow allocation that will be programmed on the physical network, i.e. $\Gamma(t+1)$. For each interaction with the digital twin, the path allocation agent applies an action that is sampled from a reduced action space, $a(n) \in \mathbb{A}^{reduced}$, where n denotes the iteration with the digital twin. Each action results in an updated path allocation for a given flow that is stored in a vector $\Gamma(n)$, and in a corresponding reward $r(n)$. To be able to compute the reward for a given allocation, the digital twin of the backhaul network models the network topology, the traffic matrix, the flow allocation, and derives an estimate of the effective data rate that would be obtained by each flow under this allocation, i.e. $\phi_i(n)$. This process is repeated for N^{steps} or until a termination condition is reached. We note that N^{steps} is a key parameter trading off the quality of the obtained allocations with the resulting execution time. This trade-off is studied in detail in Section V.

B. PHaul path computation heuristics

The goal of the path computation heuristics is to compute for each backhaul flow f_i, originating at IAB node s_i, a set of up to K^{max} potential paths through the Sub6 backhaul network and a set of up to K^{max} potential paths through the mm-wave backhaul network, where paths can be defined towards different IAB donors $d_k \in \mathbb{D}$. The output of the path selection heuristic is the set $\mathbb{P}_i^{selected}$, defined for each flow f_i. Notice that the computation of $\mathbb{P}_i^{selected}$ is performed offline, using only information about the topology of the backhaul network.

In this section we describe three path selection heuristics that are applied separately at the Sub6 and the mm-wave backhaul networks. We refer to these heuristics as the *ShortestPath*, the *LastHop* and the *LeastCommon* heuristics. These heuristics are described in Algorithm 1.

Before describing each heuristic in detail, we note that for each flow $\mathbb{P}_i^{all}$ contains a set of initial potential paths, with $|\mathbb{P}_i^{all}| > K^{max}$. The candidate flows in $\mathbb{P}_i^{all}$ can be computed using standard path computation algorithms between the source and destinations of each flow. The goal of the path selection heuristics is to select a subset of K^{max} paths from the paths available in $\mathbb{P}_i^{all}$. Considering that $\mathbb{L}$ indicates the set of links in the network, with $|\mathbb{L}| = M$, we define a path P_x as a horizontal vector $P_x = [p_1, ..., p_M]$ with $p_j = 1$ if link j belongs to path P_x, and $p_j = 0$ otherwise.

Looking at Algorithm 1, we can see that *ShortestPath* simply selects for each flow the first K^{max} shortest paths in $\mathbb{P}_i^{all}$, where paths $\mathbb{P}_i^{all}$ are already sorted in order of hops. Thus, *ShortestPath* has complexity $O(|\mathbb{F}|)$.

The idea behind *LastHop* is to select for a given flow paths that use a different last hop. The intuition behind this heuristic is that the last hop of a path is going to be reused by many flows. Therefore, allowing flows to use paths with different last hops can enhance the load balancing properties of the network. The complexity of *LastHop* is $O(|\mathbb{F}|K^{max})$, since for each flow all potential paths are evaluated.

The *LeastCommon* heuristic generalizes the idea behind *LastHop* to all links of a path. To that end, it defines a metric to compare the similarity between two paths by counting the number of common links in each path. This metric can be computed as a dot product between two path vectors, as indicated in line 32 of Algorithm 1. Thus, *LeastCommon* selects for a given flow the paths that result in the least amount of common links with respect to the already allocated paths for all the other flows. Notice that *LeastCommon* is a greedy heuristic and therefore the final path selection will depend on the order used to explore the set of flows. In our implementation *LeastCommon* starts allocating flows according to their distance from the IAB donor nodes. *LeastCommon* is the most complex heuristic with a complexity of $O(|\mathbb{F}|(\sum_{i=1}^{|\mathbb{F}|} K^{max} * i * |P_i^{all}|))$, since each flow is compared to all previously allocated flows. Notice though, that path computation heuristics need only to be run offline, and thus their complexity is not a major concern in practical networks.

Once the set $\mathbb{P}_i^{selected}$ is obtained for each flow, this set is delivered to the path allocation agent, which will consider the current traffic matrix to select the best path for each flow.

Algorithm 1: PHaul path computation heuristics.

```
Input: $\mathbb{F}$ set of all flows
Input: $K^{max}$ number of paths to select in the Sub6
       and mm-wave topologies
Input: $\mathbb{P}_i^{all}$ set of all potential paths for flow $f_i$ sorted
       in growing number of hops
Output: $\mathbb{P}_i^{selected}$ set of selected paths for flow $f_i$
1  Procedure ShortestPath()
2      for $f_i \in \mathbb{F}$ do
3          $\mathbb{P}_i^{selected} \leftarrow \mathbb{P}_i^{all}[1 : K^{max}]$
4      end
5  Procedure LastHop()
6      $\mathbb{L}^{last} \leftarrow \emptyset$
7      for $f_i \in \mathbb{F}$ do
8          for $P_x \in \mathbb{P}_i^{all}$ do
9              $last\_link \leftarrow$ last link in $P_x$
10             if $last\_link \notin \mathbb{L}^{last}$ then
11                 $\mathbb{L}^{last} \leftarrow Add(last\_link)$
                   $\mathbb{P}_i^{selected} \leftarrow Add(P_x)$
12             end
13             if $|\mathbb{P}_i^{selected}| = K^{max}$ then
14                 go to next flow
15             end
16         end
17         for $P_x \in \mathbb{P}_i^{all}$ do
18             if $P_x \notin \mathbb{P}_i^{selected}$ then
19                 $\mathbb{P}_i^{selected} \leftarrow Add(P_x)$
20             end
21             if $|\mathbb{P}_i^{selected}| = K^{max}$ then
22                 go to next flow
23             end
24         end
25     end
26 Procedure LeastCommon()
27     $\mathbb{P}_{ALL}^{selected} \leftarrow \emptyset$
28     for $f_i \in \mathbb{F}$ do
29         for $P_x \in \mathbb{P}_i^{all}$ do
30             $comn\_links \leftarrow 0$
31             for $P_y \in \mathbb{P}_{ALL}^{selected}$ do
32                 $comn\_links \leftarrow comn\_links + P_x P_y^T$
33             end
34             $\mathbb{P}_i^{unsorted} \leftarrow Add(P_x, comn\_links)$
35         end
36         $\mathbb{P}_i^{sorted} \leftarrow SortIncreasing(\mathbb{P}_i^{unsorted})$
37         $\mathbb{P}_i^{selected} \leftarrow \mathbb{P}_i^{sorted}[1 : K^{max}]$
38         $\mathbb{P}_{ALL}^{selected} \leftarrow Add(\mathbb{P}_i^{selected})$
39     end
```

C. PHaul path allocation agent

Algorithm 2 depicts the detailed operation of the PHaul path allocation agent, where the following variables are considered:

i. $NetDTwin$ is a digital twin of the wireless backhaul network, which allows to perform flow allocations and to compute the resulting per-flow effective data rates, $\phi_i(n)$, and the resulting rewards $r(n)$, (see Figure 2).

ii. The set of paths available to f_i defined as a vector of size $2K^{max}$, with the following components $\mathbb{P}_i^{selected} = \{P_{i,1}^{sub6}, P_{i,1}^{mwv}, ..., P_{i,K^{max}}^{sub6}, P_{i,K^{max}}^{mwv}\}$, where P_{sub6}^j and P_{mwv}^j represent respectively the $j-th$ path for this flow in the Sub6 or mm-wave backhaul networks.

iii. The allocation vector Γ_i, $i \le |\mathbb{F}|$, where $|\mathbb{F}|$ is the number of flows, and the $i-th$ component of this vector contains the index within $\mathbb{P}_i^{selected}$ representing the path allocated to flow f_i.

Based on the previous variables we describe now the action space, the state space and the reward used by the PHaul path allocation agent.

1) Action Space: It is defined as an integer $0 \le a \le 2K^{max}|\mathbb{F}|$, where $|\mathbb{F}|$ is the number of flows and K^{max} the number of paths available per flow in the Sub6 and mm-wave networks. Each action a encodes a single flow allocation in the following way:

i. Action a represents flow $f_i = a \mod |\mathbb{F}|$

ii. $\Gamma_i = \lfloor \frac{a}{|\mathbb{F}|} \rfloor$ indicates the index within $\mathbb{P}_i^{selected}$ corresponding to the path allocated to f_i (cf. Algorithm 2 line 22).

Thus, we can see that selecting action a represents allocating to flow $f_i = a \mod |\mathbb{F}|$ its path number $\lfloor \frac{a}{|\mathbb{F}|} \rfloor$ within the vector $\mathbb{P}_i^{selected}$. Repeated iterations of the policy will result in allocating paths for different flows.

2) State Space: Consider $\lambda_i(t)$ the input data-rate for flow f_i measured at its source $s_i \in \mathbb{S}$ in the last sampling interval, and $\phi_i(n)$, with $n < N^{steps}$, the effective data-rate for this flow measured at its destination IAB donor $d_i \in \mathbb{D}$ in the last interaction with the network digital twin[2]. Then, the environment state in PHaul is modeled as a vector containing the collection of input and effective data-rates for each flow, as well as the current path allocated to each flow (cf. Algorithm 2 line 20).

Notice that the state space in PHaul requires only the total number of IAB nodes generating backhaul flows, but it does not require detailed internal knowledge of the topology, i.e. nodes and edges. This is a design decision to make PHaul robust to internal changes on the topology, which can occur for example if a backhaul link is blocked. The PHaul state space captures the input and output traffic matrices, as well as the current paths allocated to each flow. Thus, the intuition behind the PHaul path allocation agent is that it should learn to correlate types of traffic matrixes to the specific paths that result in a good allocation for that traffic matrix, while

[2]Notice that $\phi_i(n)$ depends on the allocated path and is updated at each interaction of the PHaul agent with the network digital twin, whereas $\lambda_i(t)$ is only updated upon sampling the physical network (c.f. Figure 2)

considering the IAB network to be a black box. In Section V we present experimental results that validate this hypothesis.

3) Reward definition: The PHaul reward is modelled after the traffic engineering criteria defined by the network operator. For example, in the case of the weighted traffic engineering criteria defined in Section III, the reward $r(n)$, $n < N^{steps}$, is defined for each interaction of the agent as:

$$r(n) = (1+\gamma)\frac{\sum_i \phi_i(n)}{\sum_i \lambda_i(t)} + (1-\gamma)\frac{(\sum_i \phi_i(n))^2}{|\mathbb{F}|\sum_i \phi_i(n)^2} \quad (4)$$

Where t is the last time that the traffic matrix was sampled from the physical network. We can see that $r(n)$ encodes a balance between throughput and fairness regulated by the parameter γ. Let us therefore define the reward as:

$$r(n) = (1+\gamma)thr(n) + (1-\gamma)fair(n) \quad (5)$$

Now, given that we cannot know a priory how far from 1 both $thr(n)$ and $fair(n)$ will be in a practical network, we redefine the reward as:

$$-2 \le \hat{r}(n) = (1+\gamma)\hat{thr}(n) + (1-\gamma)\hat{fair}(n) \le 2 \quad (6)$$

$$\hat{J}(n) = \frac{(J(n) - J_{min}) - (J_{max} - J(n))}{J_{max} - J_{min}} \quad (7)$$

Where $\hat{thr}(n)$ and $\hat{fair}(n)$ are computed according to equation 7, and are constantly updated based on the rewards obtained in each interaction with the network digital twin (cf. Algorithm 2 line 3). Note that normalizing the reward to the maximum and minimum throughput or fairness values obtained throughout the N^{steps} iterations with the network digital twin is helpful for the agent to understand if the actions being applied across the different iterations are pushing the reward in the right direction.

4) Termination condition: We need at least to execute the allocation policy $N^{steps} = |\mathbb{F}|$ iterations to have the opportunity to allocate a path for each flow, and up to $N^{steps} = (2K^{max})^{|\mathbb{F}|}$ iterations, to visit all potential allocations. In practice, N^{steps} is a hyper-parameter that allows to trade-off accuracy and inference time. Section V evaluates the impact of this parameter on training accuracy. The PHaul agent terminates after executing N^{steps} iterations over the network digital twin, or as soon as an allocation is found that results in an optimal reward (cf. Algorithm 2 line 18).

5) Selected DRL algorithm: Based on the previous definitions of action space, state space and reward, any DRL agent able to operate with a discrete action space could be applied to PHaul. In this work, we base our implementation on the Proximal Policy Optimization (PPO) [6] algorithm, which achieves state-of-the-art performance across a wide range of challenging tasks and outperforms several other DRL algorithms [21]. We leave as future work the study of additional DRL policies to PHaul.

Algorithm 2: PHaul agent execution in inference mode

```
1  Procedure ComputeReward()
2      $\phi_i(n) = NetDTwin.EffectiveRates()$
3      Compute $\hat{thr}(n)$, $\hat{fair}(n)$
4      $\hat{r}(n) = (1+\gamma)\hat{thr}(n) + (1-\gamma)\hat{fair}(n)$
5      $thr_{max} = \max\{thr(n), thr_{max}\}$
6      $thr_{min} = \min\{thr(n), thr_{min}\}$
7      $fair_{max} = \max\{fair(n), fair_{max}\}$
8      $fair_{min} = \min\{fair(n), fair_{min}\}$
9  Procedure Init()
10     $n = 0$
11     $NetDTwin.Init(\{\lambda_i(t)\})$
12     $\Gamma_i = 0, \forall i \in \mathbb{F}$
13     $thr_{max} = fair_{max} = -\infty$
14     $thr_{min} = fair_{min} = \infty$
15     $NetDTwin.Alloc(\Gamma(n))$
16 Algorithm PHaul()
       Input: $\{\lambda_i(t)\}$, $NetDTwin$, $|\mathbb{F}|$, $\gamma$
       Output: $\Gamma$ vector with per-flow allocations
17     Init()
18     while $n < N^{steps}$ or $r(n) < 2$ do
19         $\phi_i(n) = NetDTwin.EffectiveRates()$
20         $state = \{\lambda_i(t), \phi_i(n), \Gamma(n)\}$
21         $a = PPO.action()$
22         $\Gamma(a \mod |\mathbb{F}|) = \lfloor \frac{a}{|\mathbb{F}|} \rfloor$
23         $NetDTwin.Alloc(\Gamma(n))$
24         ComputeReward()
25         $n = n + 1$
26     end
```

D. Visualizing a PHaul path allocation

To visualize the operation of the PHaul path allocation agent, Figure 3 depicts, on the left, the instantaneous value of the normalized reward $\hat{r}(n)$ obtained by a trained PHaul agent across $N^{steps} = 500$ iterations, and, on the right, the instantaneous action $a(n)$ being selected, where each action represents the allocation for an individual flow.

The setup considered in Figure 3 consists of a network of 17 IAB nodes and three IAB donor nodes, where each IAB node generates a backhaul flow, and $K^{max} = 1$ paths per flow are considered in the Sub6 and mm-wave backhaul networks. This results in a reduced action space of $2K^{max}|\mathbb{F}| = 34$ actions.

We see in the left part of Figure 3 the instantaneous normalized reward $\hat{r}(n)$, as well as a regression of $\hat{r}(n)$, clearly showing how the agent becomes more efficient as the number of iterations increases, where we see a saturated $\hat{r}(n) = 2$ when n approaches N^{steps}. However, recall from Algorithm 2, that PHaul does not necessarily return the allocation explored in the last iteration, but instead it will return the best performing allocation sampled across the N^{steps} iterations. It is clear from the left part of Figure 3 that N^{steps} is a critical parameter in the performance of PHaul, and as such it will be studied in detail in Section V.

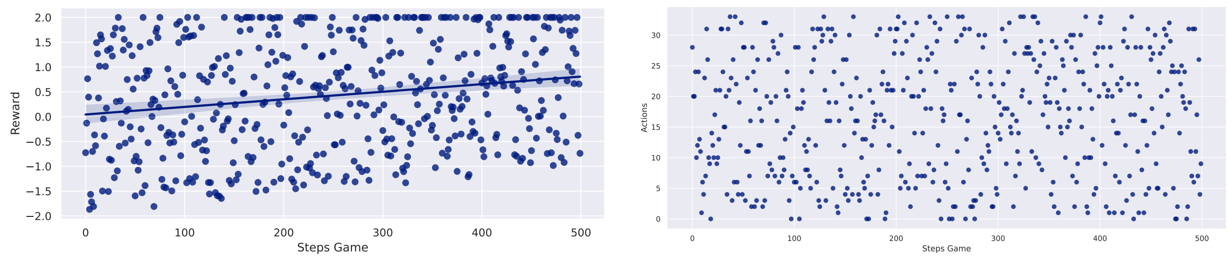


Fig. 3. Visualizing a PHaul path allocation. Left, reward $\hat{r}(n)$ across iterations with the digital twin. Right, action $a(n)$ across iterations with the digital twin.

Looking now at the right part of Figure 3, which depicts the action $a(n)$ sampled by PHaul across the iterations with the digital twin, we can see how indeed PHaul samples potential allocations for all flows. Notice that PHaul often returns to the same action, as the effect of an action will differ depending on the current allocation for all the other flows. This is why PHaul includes the current allocation as part of the state space definition.

E. Implementation discussion

To deploy PHaul in a practical 3GPP IAB system it is required to have an interface that allows the PHaul path allocation agent to dynamically update the BAP path to be used for each backhaul flow. This interface should allow to bind a backhaul flow identifier, which could be based for example on the GTP Tunnel EndPoint ID, with the BAP path identifier, which is already included in the BAP header. This interface can be implemented in the context of the O-RAN architecture by defining an E2 service model tailored to IAB networks, as already proposed in [22]. Thus, with an IAB E2 service model being available, the PHaul path allocation agent could be deployed as an xApp in the near real-time RAN Intelligent Controller (RIC), whereas the path computation heuristic and the PHaul training could be implemented as part of the non-real time RIC. Additionally, A1 policies could be used to specify the parameter γ in the xApp according to the desired traffic engineering criteria.

V. Performance Evaluation

We present in this section an in-depth performance evaluation of PHaul that is organized in the following way. In section V-A we describe our simulation setup and modelling assumptions. In section V-B we describe the competing heuristics that we use to benchmark PHaul. In section V-C we evaluate the PHaul path computation heuristics. In section V-D we optimize the training hyper-parameters of the PHaul path allocation agent, which is then benchmarked against competing approaches in section V-E. Finally, in sections V-F, V-G, and V-H we look respectively at the execution time required by the PHaul path allocation agent, at the impact of broken links, and at the ability of this agent to generalize to unseen backhaul topologies plus the gains that are due to the addition of Sub6 capabilities to the IAB backhaul.

A. Evaluation Setup

To evaluate PHaul we have developed a flow-level simulation model in Python, connected to OpenAI's Gym [23] environment, which is used to implement the PHaul path allocation agent based on PPO. We leverage the PPO implementation available in the *stable-baselines* package[3]. The developed simulator can be understood as the network digital twin component described in Section IV, i.e. it allows to generate random backhaul topologies and traffic matrices, to allocate a backhaul flow through a given path, and to estimate the effective capacity, ϕ_i, obtained by a flow under a given set of flow allocations.

To model representative IAB backhaul topologies, we developed a random topology generator after the exemplary IAB network reported in [24], which covers a suburban area in Chicago, US. In [24], the authors study different types of IAB deployments in a network consisting of up to 45 IAB nodes, where the number of donor nodes varies between 8 and 30 according to the fiber penetration in the area. Depending on the number of donors considered, the authors report chains of up to 3 hops and a mean hop count between 1.5, when the number of donors is large, and 2.08, when the number of donors is small. For our study, we are interested in scenarios where fibre penetration is low, thus we consider IAB topologies with a small number of donors, and study how these networks perform as we increase the number of non-donor IAB nodes.

We generate IAB topologies in the following way. First, we start with a first layer of 3 donor nodes. Then, for each donor node we generate n IAB child nodes, with n being a uniform random variable between 1 and 3. We repeat this process for each layer of IAB nodes until the target number of nodes in the topology is reached. All IAB nodes have a wireless link to their parent node, but we also allow for nodes to have multiple parents using a random parameter, *edge_selection_prob*, that adds an additional link between an IAB node and the IAB node located next to its parent in the upper layer. This parameter allows us to control the presence of multiple paths from a given IAB node to the wired network. Notice, that while being random, this topology generation process reflects the topology formation process in a real IAB network that would start with a fixed number of donor nodes, and then progressively grow to expand the network footprint.

[3]https://stable-baselines3.readthedocs.io/en/master/

Following our network model in Figure 1, the same IAB nodes are available in the Sub6 and mm-wave topologies, but the exact links between nodes may differ in each topology. In particular, we set *edge_selection_prob* to 0.4 in the mm-wave topology and to 0.6 in the Sub6 topology to reflect the fact that Sub6 links have wider coverage. Following this approach we consider scenarios with a total number of IAB nodes (including donor nodes) varying from 20 to 60, which result in mean hop counts of 3.08 for the case of 20 nodes and 5.12 for the case of 60 nodes.

As described in Section III, we model backhaul links as interference-free with capacities that are stable during the execution of the agent. In the case of the mm-wave backhaul, we assume a 30 GHz deployment with a 800 MHz carrier bandwidth, resulting in backhaul downlink per-link capacities in suburban environments that we select randomly between 800 Mbps and 1 Gbps, according to the simulation results reported in [25]. For the case of Sub6 links we consider an interference-free 80 MHz link with capacities that we select randomly between 200 and 300 Mbps, according to the Sub6 wireless backhaul measurements reported in [10].

Regarding the input traffic matrix, we consider that each IAB node aggregates the traffic from all connected UEs into a single IAB backhaul flow. Thus, the traffic carried by this flow will depend on the simultaneous number of UEs in RRC_CONNECTED state in each IAB node, and on the capacity of the access interface. Lacking this type of data from real deployments, we model the load offered by each backhaul flow in the following way. First, we use a random parameter, *node_active_probability*, to determine if there are active UEs, i.e. in RRC_CONNECTED mode, in that node. In case there are active UEs, we model the resulting backhaul traffic as a uniform random variable between λ_{min} and λ_{max}, where we consider two scenarios with a growing flow size in Mbps, namely: i) $\lambda_{min} = 250$, $\lambda_{max} = 500$, and ii) $\lambda_{min} = 500$, $\lambda_{max} = 750$.

Finally, to achieve statistically significant results, every time we report a performance figure for a given network configuration, we consider at least 10 random topologies for that network configuration, and for each of those topologies we average the results of 250 randomized input traffic matrixes. Unless otherwise stated, the PHaul path allocation agent is trained for each specific topology considered. The metrics reported in this section correspond to average values that are depicted with their corresponding 95% confidence interval, which is however too small to be clearly seen in the figures.

To ease the reproducibility of the results presented in this section, we have released as open source our implementation of PHaul, the implementation of the IAB network digital twin, and all the supporting evaluation environments[4].

B. Competing heuristics

To assess the performance of PHaul, we focus on two metrics: i) the score achieved in terms of the weighted objective function J^{TE} introduced in Section III, and ii) the required execution time. Recall that the weighted objective function is defined as $J^{TE} = (1+\gamma)\frac{\sum_i \phi_i(t)}{\sum_i \lambda_i(t)} + (1-\gamma)\frac{(\sum_i \phi_i(t))^2}{|\mathbb{F}|\sum_i \phi_i(t)^2}$, where we can set $\gamma = 1$ to focus exclusively on throughput efficiency, hereafter referred to simply as efficiency, and $\gamma = -1$ to focus on inter-flow fairness.

PHaul is then compared to three alternative path allocation agents, namely: i) *brute force*, ii) *subset-sum*, and iii) *random*. All the evaluated path allocation agents use the same path selection heuristic (c.f. Section IV).

The *brute force* path allocation agent considers, in the digital twin, all combinations between flows and paths and selects the one yielding the highest reward $\hat{r}(n)$. Thus, for a given number of flows $|\mathbb{F}|$ and paths K^{max}, brute force explores $(2K^{max})^{|\mathbb{F}|}$ possible allocations in the network digital twin. For every considered allocation, brute force evaluates the objective function J^{TE}, with γ set accordingly to maximize efficiency or fairness. Brute force is optimal in terms of the objective function, but it leads to large execution times and it can only be used for network configurations with a limited number of flows and paths.

The *subset-sum* path allocation agent is a greedy heuristic based on the subset-sum problem [26]. Subset-sum orders backhaul flows in decreasing order of their traffic demand λ_i, and allocates each flow sequentially. For each flow, the agent explores allocations in the $2K^{max}$ paths available to that flow, and selects the one that maximizes the objective function J^{TE}, with γ set accordingly. Subset-sum scales linearly with the number of flows $|\mathbb{F}|$, evaluating up to $2K^{max}|\mathbb{F}|$ allocations.

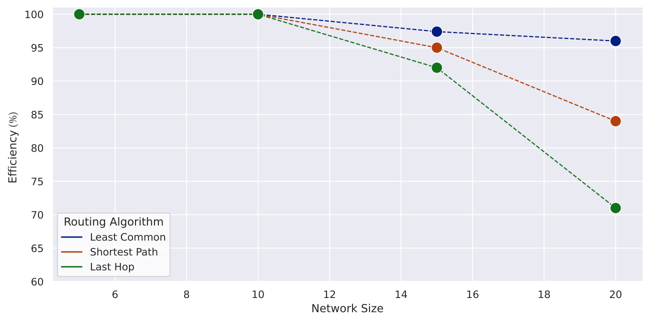
Finally, the *random* path allocation agent, simply selects for each flow one of the $2K^{max}$ paths available through the mm-wave and Sub6 backhaul networks using a uniform random variable, scaling linearly with the number of flows $|\mathbb{F}|$.

C. Evaluating the PHaul path computation heuristics

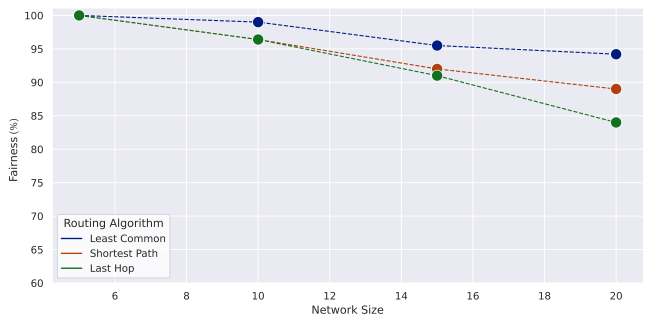
We start our evaluation by looking at the path computation heuristics described in Section IV-B. To isolate the effect of the path computation heuristics from the impact of the path allocation agent, we evaluate the different path computation heuristics using the brute force path allocation agent. We note that using the brute force path allocation agent is not possible in practice, but our goal in this experiment is to compare path computation heuristics under a common setting, whithout the influence of the PHaul agent coming into play. However, using the brute force path allocation agent severely limits the size of the topologies that we can test. Hence, we limit this evaluation to network sizes of up to 20 IAB nodes (including the 3 donor nodes), and only $K^{max} = 1$ paths are selected for each flow in the Sub6 and the mm-wave networks. To stress the network, we set *node_active_probability* to 1 and consider an input traffic matrix characterized by $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps.

Figure 4 depicts the performance of the *ShortestPath* (orange), *LastHop* (green) and *LeastCommon* (blue) path computation heuristics on throughput efficiency (left graph), with $\gamma = 1$ in J^{TE}, and fairness (right graph), with $\gamma = -1$ in J^{TE}.

[4]https://github.com/Fundacio-i2CAT/phaul/



(a) Efficiency ($\gamma = 1$) for $K^{max} = 1$



(b) Fairness ($\gamma = -1$) for $K^{max} = 1$

Fig. 4. Impact of PHaul path computation heuristics assuming a brute-force forwarding agent

We can see in Figure 4 how the *LeastCommon* heuristic clearly outperforms the *ShortestPath* and *LastHop* heuristics both in terms of throughput efficiency and fairness. The performance of the different path computation heuristics is affected by the IAB topologies considered in our evaluation, which, as previously described, model realistic deployments that tend to follow a tree-like structure that arises naturally when the IAB network is built progressively starting from the donor nodes. Under this type of network topology, flows originating at a given IAB node may share links in their path with all IAB flows that originate at higher levels of the topology. Therefore, heuristics like *ShortestPath* or *LastHop*, which focus only on the path allocations for a single flow at a time, result in links being reused by many flows. On the other hand, *LeastCommon*, by considering the path allocations of previous flows and minimizing the number of joint links across flows, is able to select paths that are more disjoint, resulting in an increased effective capacity in the network.

We hereafter use the *LeastCommon* heuristic as our default path computation heuristic.

D. Training the PHaul path allocation agent

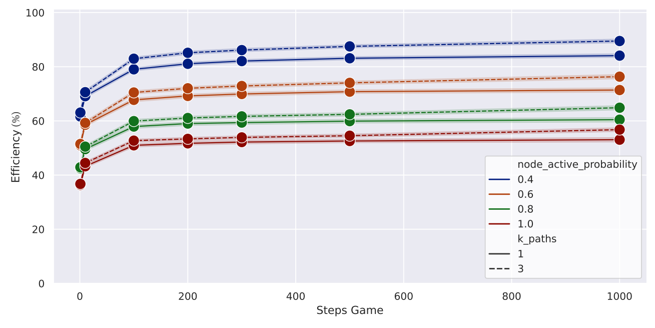
We evaluate in this section the training phase of the PHaul path allocation agent. Our goal is twofold. First, we want to understand how PHaul's performance depends on our training hyper-parameters, namely N^{steps} used in Algorithm 2, and the parameter *training_steps*, which determines the overall number of steps considered in the training phase of PPO [6]. Second, we want to benchmark the performance gap between PHaul and the brute force agent, which provides an optimal performance. As performance metrics, we look separately at efficiency obtained with $\gamma = 1$ in J^{TE}, and then at fairness, obtained with $\gamma = -1$ in J^{TE}.

Figure 5 depicts in the upper row the impact of N^{steps} while fixing *training_steps* to 10^5, and in the lower row the impact of *training_steps* while fixing N^{steps} to 300. In these experiments, we consider an IAB network size of 50 nodes, and vary the *node_active_probability* parameter between 0.4 and 1, while considering a random backhaul flow rate between $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps. We depict experiments for $K^{max} = 1$ and $K^{max} = 3$ paths in the Sub6 and mm-wave networks.

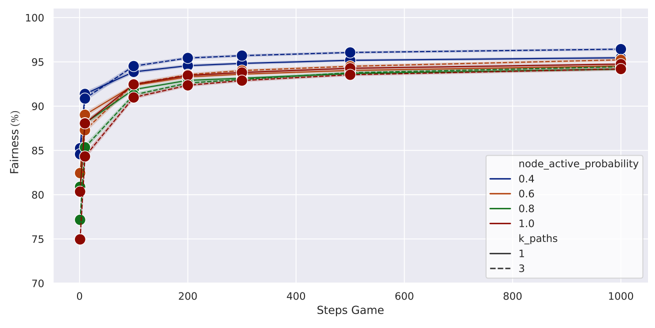
We can see how, both for efficiency and fairness, performance is low when N^{steps} and *training_steps* are small, and smoothly increases when these parameters grow. Notice though that N^{steps} will have an impact on the execution time of a trained agent, while *training_steps* only affects the overall training time, which is not a critical parameter. Looking at the knee exhibited by the training curves, we can see that this is independent from the *node_active_probability* parameter, which means that regardless of the level of activity in the network the PHaul agent training performance is maintained. The number of paths $K^{max} = 1$ and $K^{max} = 3$ does not impact either the position of the knee in the training curves. Based on these results the training of PHaul agent can be simplified by setting a fixed value of N^{steps} and *training_steps*, regardless of the number of paths or the amount of active flows, which relaxes the requirements to train the PHaul network digital twin in real networks. Hereafter we consider $N^{steps} = 300$ and $training\_steps = 20000$.

We observe on the left part of Figure 5 how efficiency decreases when increasing *node_active_probability*. This is expected because a higher *node_active_probability* means more load being injected into the wireless backhaul, which is saturated in all cases. The impact of *node_active_probability* is however not so clear when looking at fairness (right part of Figure 5). The reason is that the regardless of the number of backhaul flows active in the network, PHaul is able to allocate the bottleneck bandwidth across these flows in a fair way. Looking at the impact of varying the number of paths K^{max} from 1 to 3, as expected, we observe that considering more paths leads to higher network efficiency as flows can be better balanced through the network. Notice though, that the potential gain achieved by increasing K^{max} depends on the path diversity available in the IAB topology, which is limited in our scenarios that mostly consist in tree-like topologies with limited multi-path opportunities.

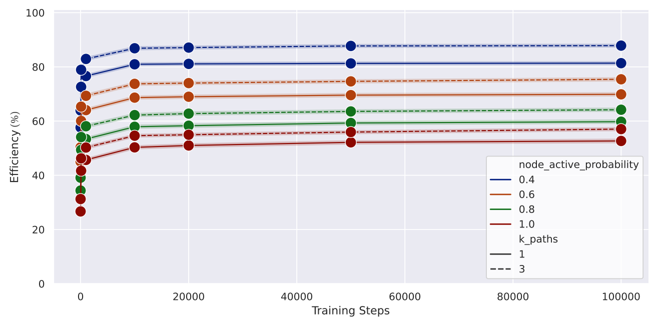
To assess how far the PHaul performance lies from the optimal allocation, we compare in Figure 6 PHaul against the brute-force agent in terms of efficiency ($\gamma = 1$) and fairness



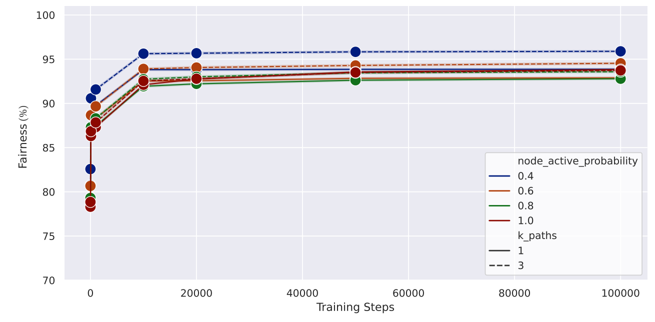
(a) Impact of N^{steps} on efficiency with $training\_steps = 10^5$



(b) Impact of N^{steps} on fairness with $training\_steps = 10^5$



(c) Impact of $training\_steps$ on efficiency with $N^{steps} = 300$



(d) Impact of $training\_steps$ on fairness with $N^{steps} = 300$

Fig. 5. Training against Brute Force

($\gamma = -1$), considering $N^{steps} = 300$ and $training\_steps = 20000$. Due to the high computational requirements of the brute force agent we are only able to carry out this benchmark with a limited network size of 20 IAB nodes and with $K^{max} = 1$. To have a meaningful comparison, we need to ensure that the network is saturated, for which we configure *node_active_probability*$= 1$ and use flow sizes with $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps. We can see in Figure 6 that both the efficiency and fairness achieved by PHaul lie very close to the brute force agent, which validates the performance of PHaul in the considered scenario. The performance gap between PHaul and brute force is however expected to increase if larger topologies are considered.

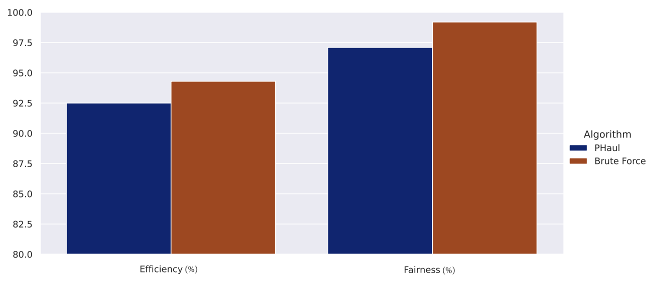


Fig. 6. PHaul vs Brute Force for network size 20 and *node_active_probability*$= 1$

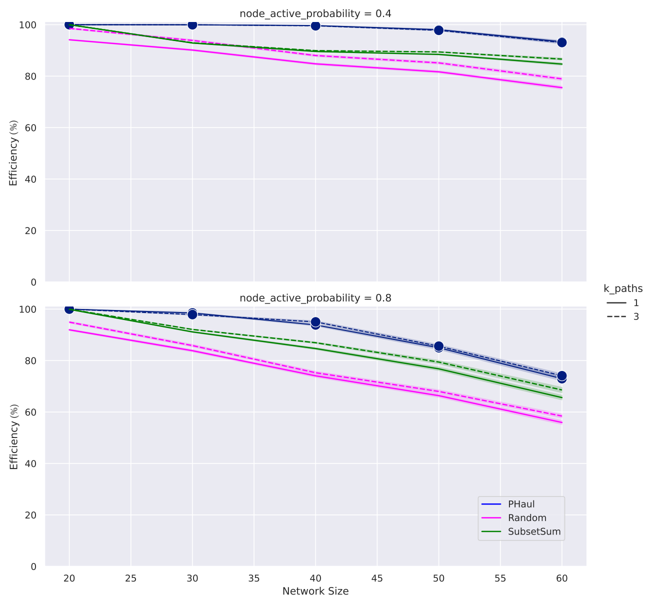
After proving that the PHaul path allocation agent can be effectively trained towards different objective functions, we evaluate next its performance against the subset-sum and radom path allocation agents for a variety of network sizes.

E. Inference mode evaluation against competing heuristics

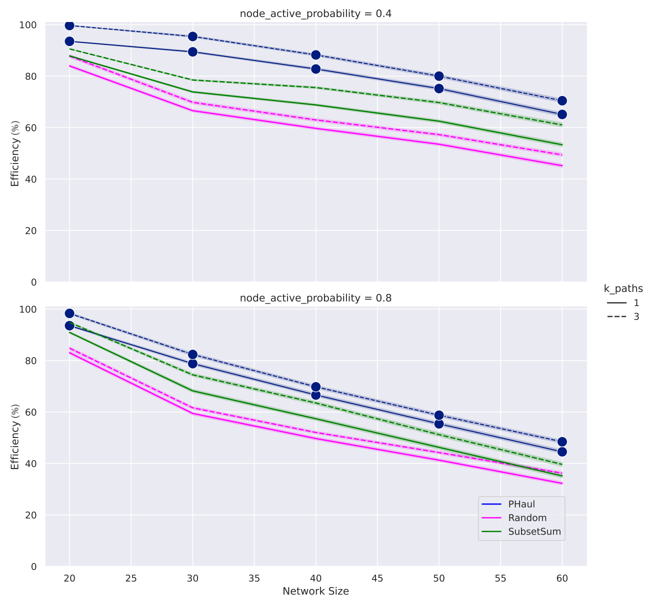
Next, we evaluate the performance of the PHaul, the subset-sum and the random path allocation agents, when increasing the size of the IAB network from 20 to 60 nodes, including 3 IAB donor nodes. We do not consider the brute force agent in this section because of its excessive computational time. To consider different load conditions in the network, we set *node_active_probability* to 0.4 and 0.8 and we evaluate two types of input traffic matrix distributions with a growing flow size in Mbps, namely: i) $\lambda_{min} = 250$, $\lambda_{max} = 500$, and ii) $\lambda_{min} = 500$, $\lambda_{max} = 750$. All agents are evaluated for $K^{max} = 1$ path in the mm-wave and Sub6 networks, and for $K^{max} = 3$ paths in the mm-wave and Sub6 networks.

Figure 7 depicts efficiency ($\gamma = 1$) when the network size grows between 20 and 60 IAB nodes, with *node_active_probability* $= 0.4$ in the upper row, *node_active_probability* $= 0.8$ in the lower row, flow size in Mbps between $\lambda_{min} = 250$ and $\lambda_{max} = 500$ in the left column and flow size in Mbps between $\lambda_{min} = 500$ and $\lambda_{max} = 750$ in the right column.

We can see how for all network configurations the PHaul agent outperforms subset-sum, which in turn outperforms the random agent. A maximum gain of 17% is observed for PHaul when compared to subset-sum, and of 36% when compared to random. All agents benefit from considering a larger number



(a) Flow size $\lambda_{min} = 250$ Mbps, $\lambda_{max} = 500$ Mbps



(b) Flow size $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps

Fig. 7. Efficiency ($\gamma = 1$) with a growing network size

of paths, but this gain is more evident when the flow data rates are higher ($\lambda_{min} = 500$, $\lambda_{max} = 750$). Note that subset-sum is a well proven bin-packing heuristic that sorts backhaul flows in decreasing order of size and greedily starts allocating them one at a time. The reason why PHaul is able to outperform subset-sum, is that in the training process PHaul is able to learn a representation of the topology of the IAB network, which it can then correlate with a given traffic matrix distribution to derive non-trivial allocations that result in good performance. For all the agents efficiency decreases as network size increases, and the decrease is higher for higher flow data rates. The reason for this behavior is that introducing new IAB nodes results in a higher offered load, regulated by the parameter *node_active_probability*. This effect dominates over any increase in cross-section bandwidth that may result from the additional backhaul links contributed by the new IAB nodes.

Figure 8 depicts now the results in terms of fairness ($\gamma = -1$), for the same experiments described in Figure 7. Unlike efficiency, fairness exhibits a rather flat behavior when the size of the IAB network grows. When the IAB network grows, the number of backhaul flows increases and the effective bandwidth allocated to each flow decreases. In our network model, each IAB node allocates per-flow capacities in a bottleneck link using a water-filling algorithm. Therefore, the means that the different agents have to improve fairness is to select the paths for each flow such that all flows in the network achieve a similar effective capacity. We can see in Figure 8 how PHaul is the best agent in achieving a fair allocation, resulting in a close to perfect fairness for all considered network settings, and for both 1 and 3 available paths. A maximum gain of 13% is observed in terms of fairness for PHaul when compared to subset-sum, and of 20% when compared to random. The reason for the good performance of PHaul, is that in the training phase PHaul is able to learn correlations between a given traffic matrix and the set of available paths that result in a good fairness metric.

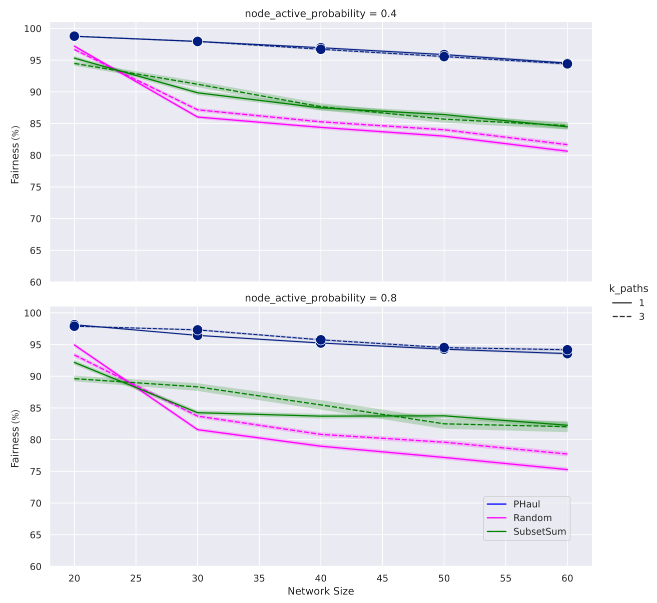
After verifying in this section that PHaul consistently offers a superior performance in terms of throughput efficiency and fairness than subset-sum and random, we study next the performance of the different agents in terms of execution time.

F. Benchmarking PHaul Execution Time

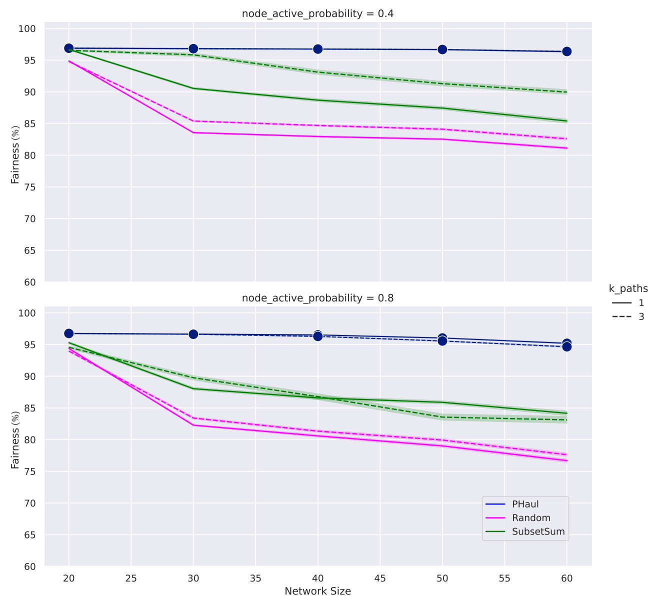
The main goal of PHaul, as stated in Section IV, is to periodically read the traffic matrix from the physical network to then update the mapping between backhaul flows and pre-provisioned backhaul paths. The frequency of these updates is thus limited by the execution time of the path allocation agent.

Figure 9 depicts the average execution time of the PHaul, subset-sum and random agents, when increasing the IAB network between 20 and 60 nodes. The left hand size depicts the execution times when optimizing for efficiency, which result from averaging all the network configuration considered in Figure 7. The right hand side depicts the corresponding execution time when optimizing for fairness, considering the network configurations included in Figure 8.

The execution times reported Figure 9 are of course dependent on the platform where the agents are run, which in our case consist of a single Intel Xeon E5-2618L v4 CPU. Notice that in a real implementation the PHaul path allocation agent can be implemented in a centralized location where enough compute resources are available.

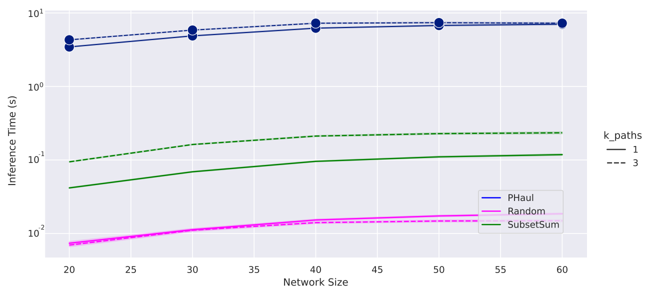


(a) Flow size $\lambda_{min} = 250$ Mbps, $\lambda_{max} = 500$ Mbps

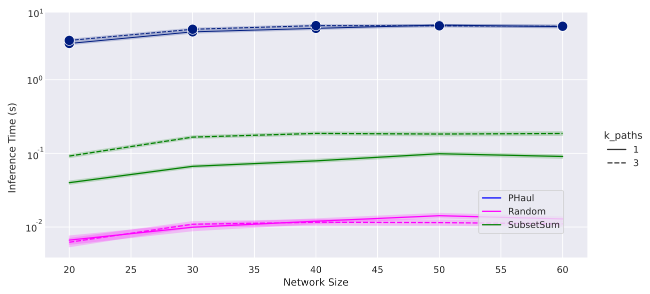


(b) Flow size $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps

Fig. 8. Fairness ($\gamma = -1$) with a growing network size



(a) Efficiency ($\gamma = 1$) for $K^{max} = 1$ and $K^{max} = 3$



(b) Fairness ($\gamma = -1$) for $K^{max} = 1$ and $K^{max} = 3$

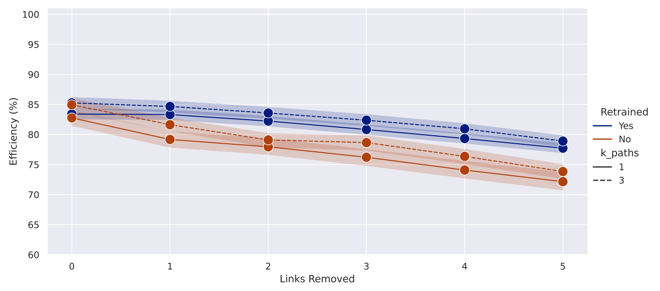
Fig. 9. Inference time (secs)

We can see in Figure 9 how, as expected, the execution times of PHaul are larger than those of subset-sum and random. However, PHaul keeps an execution time below 10 seconds that slightly increases with the network size. A 10 second interval to reconfigure the forwarding in the backhaul is a reasonable value because UEs connected to a cell transition from RRC_CONNECTED to RRC_IDLE after an inactivity timeout of around 10 seconds [11], which means that sampling at this frequency is adequate to observe significant changes in the traffic matrix.

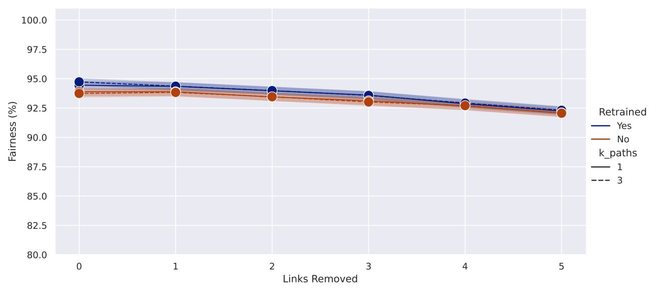
It is also relevant to observe how the execution time of PHaul is fairly independent of the network size. The reason is that the execution time of PHaul is dominated by the N^{steps} parameter, which defines the number of interactions with the network digital twin (c.f. Section IV). It is thus possible to reduce execution time in PHaul by reducing N^{steps}, at the cost of losing accuracy in terms of the objective function, as depicted in Figure 5. Regarding the number of paths K^{max}, we can see that they only have a marginal impact on the execution time of PHaul. The reason is that increasing K^{max} translates into an increase in the size of the action space, which may impact convergence time in the training phase, but it results in a minor impact with respect to the time required to decide what action to choose in inference phase. This is not the same for subset-sum, which is clearly affected by the number of paths, as it needs to evaluate $2K^{max}$ candidate path allocations for each flow. Finally, the execution time of the random agent is negligible, as it only involves computing a random number.

G. Impact of broken links

Given the nature of the IAB wireless backhaul, transitory broken links can occur due to link blockage at mm-wave frequencies, or due to unplanned interference at Sub6 frequencies. The goal of this section is to evaluate how resilient is PHaul to these events, since it is not realistic to assume that PHaul can be retrained every time a link breaks. Upon



(a) Efficiency ($\gamma = 1$) for $K^{max} = 1$ and $K^{max} = 3$



(b) Fairness ($\gamma = -1$) for $K^{max} = 1$ and $K^{max} = 3$

Fig. 10. Impact of retraining when removing links from the network, for flow size ($\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps) and $node\_active\_probability = 0.4$

a broken link, the internal IAB routing protocol will restore end-to-end connectivity by re-routing backhaul flows. Notice that the PHaul action space simply assigns for each flow a path defined by a path index. Thus, if upon a link break the actual hop-by-hop sequence of a path is modified, but the path index is maintained and the path still connects the same source and destination, then PHaul is able to continue forwarding packets through that path. Nevertheless, the resulting topology upon a link break will differ from the topology that PHaul has been trained on, and hence a performance degradation can be expected. The goal of this section is to quantify this degradation.

Figure 10 depicts the results of an experiment where we consider a network of 40 IAB nodes, with flow size $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps and $node\_active\_probability = 0.4$, and increase the number of simultaneous broken links from 1 to 5, which we consider representative of this network size. For each point in the x-axis we consider 10 different topologies and 250 random samples with different traffic matrixes. In each sample we randomly remove x links from the network, but ensure that end-to-end connectivity for all backhaul flows remains possible. Figure 10 compares the performance in terms of efficiency ($\gamma = 1$) and fairness ($\gamma = -1$), considering the ideal performance where PHaul is retrained every time that a link is removed from the topology (shown in blue), versus the performance to be expected in practice when PHaul has only been trained for the full topology but continues to perform inferences when links are removed (shown in brown).

Looking at Figure 10(a) we see that overall efficiency reduces as the number of broken links increases, because the network has less capacity, but the efficiency loss because of having PHaul operate over a network with broken links is only around 5%, both for $K^{max} = 1$ and $K^{max} = 3$ paths. Looking at Figure 10(b) we observe that the loss in fairness is even smaller, being around 2%. We note that the tree-like structure of IAB backhaul networks tends to result in backup paths that are similar to the original ones, which helps explain the good performance of PHaul observed in this experiment.

In the next section we continue analyzing the resilience of PHaul to untrained topologies that significantly differ from the topology PHaul has been trained on.

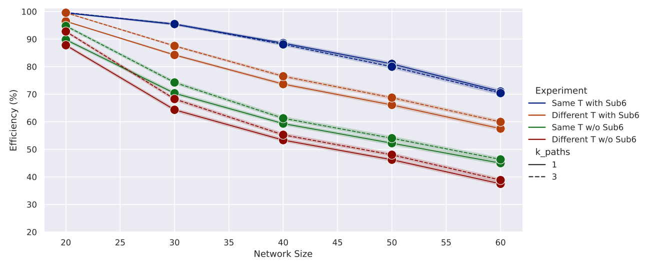
H. Impact of untrained topologies and Sub6 spectrum

Having seen in the previous section that PHaul is reliable to small changes in the topology, in this section we deepen our evaluation in two directions. First, we evaluate the performance of PHaul on topologies that differ significantly from the topology that PHaul has been trained on. Second, we evaluate the performance of PHaul with and without Sub6 connectivity, to quantify the gains that adding Sub6 spectrum provides on efficiency and fairness metrics.

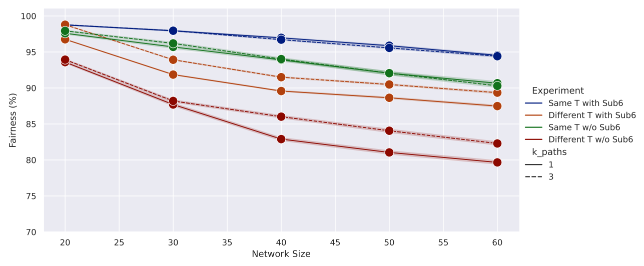
To evaluate PHaul with untrained topologies, we consider again 10 topologies for each considered network size, but train PHaul with only one of the 10 topologies. All topologies share the same number of IAB nodes and the same number of donor nodes, which is what makes PHaul applicable across them. Topologies though, differ in the number of layers and in the links connecting IAB nodes. To evaluate the gains provided by Sub6 spectrum, we run each topology under two configurations: i) a first configuration where both mm-wave and Sub6 spectrum are available, and ii) a second configuration where only mm-wave links are available.

Figure 11 depicts for a growing network size the performance in terms of efficiency with $\gamma = 1$ (left graph) and fairness with $\gamma = -1$ (right graph). For this experiment we assume flow size $\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps and $node\_active\_probability = 0.4$, and consider $K^{max} = 1$ and 3. We consider the following four configurations: i) Sub6 enhanced IAB where PHaul is trained specifically for each topology (blue lines), ii) Sub6 enhanced IAB where PHaul is only trained for one topology (brown lines), iii) mm-wave only IAB where PHaul is trained for each topology (green lines), and iv) mm-wave only IAB where PHaul is only trained for one topology (red lines). For each configuration we consider $K^{max} = 1$ and $K^{max} = 3$ paths.

Looking at the impact in efficiency in Figure 11(a), we can see how using untrained topologies results in a slight degradation of about 9% in efficiency. Removing Sub6 spectrum, but retraining PHaul every time, results in an efficiency loss of around 25%, which justifies the gains provided by Sub6 spectrum, since, using PHaul, a Sub6 enhanced IAB network



(a) Efficiency ($\gamma = 1$) for $K^{max} = 1$ and $K^{max} = 3$



(b) Fairness ($\gamma = -1$) for $K^{max} = 1$ and $K^{max} = 3$

Fig. 11. Impact of untrained topologies and Sub6 spectrum, for flow size ($\lambda_{min} = 500$ Mbps, $\lambda_{max} = 750$ Mbps) and $node\_active\_probability = 0.4$

with untrained topologies is still more efficient that a perfectly trained mm-wave only IAB network. Finally, removing Sub6 spectrum and using untrained topologies results in an overall degradation slightly above 30%.

Looking at fairness in Figure 11(b) we can see that the trend is maintained, but interestingly, removing Sub6 spectrum results in a degradation of around 4%, whereas using untrained topologies result in a higher degradation of around 7%. The reason is that when removing Sub6 spectrum the overall network capacity reduces, but PHaul is still able to allocate the available capacity across flows in a fair way. Fairness is however impacted when PHaul runs over untrained topologies, although the impact is small. Finally, removing Sub6 spectrum and using untrained topologies results in the worst case ($K^{max} = 1$) in a degradation of around 15%.

These results quantify the benefits of adding Sub6 spectrum to the IAB backahul, as proposed in this paper, and validate that PHaul has a graceful degradation when used over topologies that differ significantly from the topologies that have been used for training, which validate the application of PHaul in practical networks.

VI. Conclusions

Future bandwidth-hungry 6G services will require the deployment of wireless access networks providing gigabit capacities. A key challenge towards the design of 6G is therefore the development of network architectures that can provide the required capacity while being economically sustainable. 3GPP IAB is a novel network architecture that addresses this concern by allowing to reuse the same spectrum in access and backhaul, thus minimizing the need of expensive fibre layouts when deploying a new radio access network.

In this paper we have proposed to enhance 3GPP IAB networks by adding Sub6 spectrum in the backhaul segment, where the greenfield 6 GHz band, validated in WRC-23 for IMT use [5], could be used for this purpose. In addition to the added backhaul capacity, the Sub6 spectrum complements mm-wave in terms of coverage, which is critical in dense urban environments. Based on our Sub6 enhanced IAB architecture, we have proposed PHaul, which is a novel DRL-based forwarding agent for IAB networks that adapts the paths used to forward the backhaul flows according to a periodically sampled traffic matrix.

Using a simulation approach that models realistic IAB topologies, we have analyzed the training properties of PHaul and have evaluated its performance in terms of throughput efficiency and fairness with respect to two competing IAB forwarding agents. In our experiments PHaul always outperforms competing approaches, demonstrating gains of up to 36% in terms of efficiency and of up to 20% in terms of fairness. We have shown that the PHaul is able to execute path allocation inferences in less than 10 seconds, which is a reasonable frequency to be applied to real IAB networks. Finally, we have shown that PHaul degrades smoothly when there are differences between the IAB topology and the topology that PHaul has been trained on, and we have explicitly quantified the performance gain that is due to adding Sub6 capabilities to the IAB network.

As future work we consider the comparison of different DRL algorithms applied to the PHaul forwarding agent, as well as the evaluation of additional traffic engineering criteria, for example incorporating criteria related to energy efficiency.

References

[1] W. Chen et al., ”5G-Advanced Toward 6G: Past, Present, and Future,” in IEEE Journal on Selected Areas in Communications, vol. 41, no. 6, pp. 1592-1619, June 2023, doi: 10.1109/JSAC.2023.3274037.

[2] M. Polese, J. M. Jornet, T. Melodia and M. Zorzi, ”Toward End-to-End, Full-Stack 6G Terahertz Networks,” in IEEE Communications Magazine, vol. 58, no. 11, pp. 48-54, November 2020, doi: 10.1109/MCOM.001.2000224.

[3] TS 38.300 V16.8.0, NR and NG-RAN Overall Description; Stage 2 (Release 16),” December 2021

[4] O. Semiari, W. Saad, M. Bennis and M. Debbah, ”Integrated Millimeter Wave and Sub-6 GHz Wireless Networks: A Roadmap for Joint Mobile Broadband and Ultra-Reliable Low-Latency Communications,” in IEEE Wireless Communications, vol. 26, no. 2, pp. 109-115, April 2019, doi: 10.1109/MWC.2019.1800039.

[5] GSMA, ”GSMA hails groundbreaking spectrum decisions at WRC-23”. Available online: https://www.gsma.com/newsroom/press-release/gsma-hails-groundbreaking-spectrum-decisions-at-wrc-23/

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). Proximal policy optimization algorithms.

[7] K. Teramae, K. Mizutani, T. Matsumura and H. Harada, ”Enhancement of User Perceived Throughput in Sub-6 GHz Integrated Access and Backhaul with Dynamic Full-Duplex,” 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), Honolulu, HI, USA, 2019, pp. 1-7, doi: 10.1109/VTCFall.2019.8891086.

[8] Mengxin Yu, Yibo Pi, Aimin Tang, Xudong Wang, ”Coordinated parallel resource allocation for integrated access and backhaul networks”, Computer Networks, Volume 222, 2023, https://doi.org/10.1016/j.comnet.2022.109533.

[9] M. Diamanti, P. Charatsaris, E. E. Tsiropoulou and S. Papavassiliou, ”The Prospect of Reconfigurable Intelligent Surfaces in Integrated Access and Backhaul Networks,” in IEEE Transactions on Green Communications and Networking, vol. 6, no. 2, pp. 859-872, June 2022, doi: 10.1109/TGCN.2021.3126784

[10] Camps-Mur, D., Gutierrez, J., Grass, E., Tzanakaki, A., Flegkas, P., Choumas, K., ... and Simeonidou, D. (2019). 5G-XHaul: A novel wireless-optical SDN transport network to support joint 5G backhaul and fronthaul services. IEEE Communications Magazine, 57(7), 99-105.

[11] A. Betzler, D. Camps-Mur, E. Garcia-Villegas, I. Demirkol and J. J. Aleixendri, ”SODALITE: SDN Wireless Backhauling for Dense 4G/5G Small Cell Networks,” in IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1709-1723, Dec. 2019, doi: 10.1109/TNSM.2019.2930745.

[12] Amaldi, E., Capone, A., Coniglio, S., & Gianoli, L. G. (2013). Network optimization problems subject to max-min fair flow allocation. IEEE communications letters, 17(7), 1463-1466.

[13] Reis, J., Phan, T. K., Kheirkhah, M., Yang, F., Griffin, D., Rocha, M., & Rio, M. (2021, July). R2L: routing with reinforcement learning. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.

[14] Mellouk, Abdelhamid, Said Hoceini, and Samia Larynouna. ”Flow based routing for irregular traffic using reinforcement learning approach in dynamic networks.” 11th IEEE Symposium on Computers and Communications (ISCC’06). IEEE, 2006.

[15] Chilukuri, Shanti, and Dirk Pesch. ”RECCE: Deep reinforcement learning for joint routing and scheduling in time-constrained wireless networks.” IEEE Access 9 (2021): 132053-132063.

[16] Yin, Hao, Sumit Roy, and Liu Cao. ”Routing and resource allocation for IAB multi-hop network in 5G advanced.” IEEE Transactions on Communications 70.10 (2022): 6704-6717.

[17] Madapatha, Charitha, et al. ”On topology optimization and routing in integrated access and backhaul networks: A genetic algorithm-based approach.” IEEE Open Journal of the Communications Society 2 (2021): 2273-2291.

[18] ”FCC Opens 6 GHz Band to Wi-Fi and Other Unlicensed Uses”. Available online: https://www.fcc.gov/document/fcc-opens-6-ghz-band-wi-fi-and-other-unlicensed-uses-0

[19] Radunovic, Bozidar, and Jean-Yves Le Boudec. ”A unified framework for max-min and min-max fairness with applications.” IEEE/ACM Transactions on networking 15.5 (2007): 1073-1083.

[20] Massoulié, Laurent, and James Roberts. ”Bandwidth sharing: objectives and algorithms.” IEEE INFOCOM’99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320). Vol. 3. IEEE, 1999.

[21] Wang, Yuhui and He, Hao and Wen, Chao and Tan, Xiaoyang (2019). Truly Proximal Policy Optimization.

[22] Moro, E., Gemmi, G., Polese, M., Maccari, L., Capone, A., and Melodia, T. (2023, June). Toward open integrated access and backhaul with O-RAN. In 2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet) (pp. 61-69). IEEE.

[23] Brockman, Greg and Cheung, Vicki and Pettersson, Ludwig and Schneider, Jonas and Schulman, John and Tang, Jie and Zaremba, Wojciech (2016). OpenAI Gym.

[24] M. Cudak, A. Ghosh, A. Ghosh and J. Andrews, ”Integrated Access and Backhaul: A Key Enabler for 5G Millimeter-Wave Deployments,” in IEEE Communications Magazine, vol. 59, no. 4, pp. 88-94, April 2021, doi: 10.1109/MCOM.001.2000690.

[25] Ronkainen, H., Edstam, J., Ericsson, A., & Östberg, C. (2020). Integrated access and backhaul a New Type of Wireless Backhaul in 5G. Ericsson Technology Review, 2020(7), 2-11.

[26] Kleinberg, Jon; Tardos, Éva (2006). Algorithm Design (2nd ed.). p. 491. ISBN 0-321-37291-3.

Jorge Pueyo is a research and development engineer at i2CAT (Barcelona, Spain). He holds a Bachelor’s degree in Telecommunication Technologies and Services Engineering from the Polytechnic University of Catalonia (UPC) and a Master’s degree in Advanced Telecommunications Technologies, also from UPC. He has extensive experience in software development, primarily using Python and Java, as well as expertise in cloud development, data analysis, and machine learning. Currently, he is actively involved in various European projects related to O-RAN technologies.

Daniel Camps-Mur is currently leading the Mobile and Wireless Internet (MWI) group at I2CAT in Barcelona, Spain. Previously, Daniel was a senior researcher at NEC Network Laboratories in Heidelberg, Germany. In 2004 he received a Master’s degree and in 2012 a Ph.D. degree from the Polytechnic University of Catalonia (UPC). His research interests include mobile networks, software defined networking and communications protocols for the Internet of Things.

Miguel Catalan-Cid is a senior research engineer at i2CAT in Barcelona, Spain. He holds since 2008 a Master’s degree and since 2016 a Ph.D. from the Polytechnic University of Catalonia (UPC). He has an extensive experience in wireless networks, analysis and definition of communication protocols, utilisation of simulation tools, and programming embedded systems and micro-controllers. He is currently being involved in different H2020 projects related to B5G and O-RAN technologies.