COLA: Combinatorial Optimization Learning Approach for Energy-Efficient Backhauling in UAV-Assisted B5G/6G Wireless Networks

Oluwaseun T. Ajayi, *Student Member, IEEE,* Suyang Wang, *Member, IEEE*, and Yu Cheng, *Fellow, IEEE*

***Abstract*—The ultra densification of small cells to meet the heterogeneous traffic demands of users is a major attribute of the beyond 5G (B5G)/6G wireless networks which is still being studied. Unmanned aerial vehicle (UAV) base stations (UBSs) which can grasp tall buildings and lampposts offer flexible deployment opportunities to the B5G/6G wireless network more than do fixed base stations, as they serve as relay nodes to provide wireless backhauling for ultra-reliable low latency communications. However, the key challenge lies in jointly optimizing the UBSs deployment and multi-hop wireless backhauling to minimize the overall energy consumption. This involves finding the shortest paths with minimum energy costs, which is a mixed integer linear programming problem that is NP-hard. To tackle this problem, we propose a combinatorial optimization learning approach (COLA) that integrates a combinatorial algorithm into a neural network to quickly and effectively approximate the solution to the optimization problem. First, the neural network predicts the "magic" link weights given the application-level inputs. The link weights are then used by a simple Dijkstra's algorithm to find the shortest paths for the multi-commodity flows. We will demonstrate how to decompose the original multi-hop multi-commodity flow optimization problem into learnable single-commodity shortest path problems, how to obtain the training data, and how to backpropagate the gradients incorporating the combinatorial component during training. We evaluate COLA in both small-sized and large-sized wireless networks. Our COLA greatly reduces the computation overhead, while maintaining near-optimal solution quality in terms of the energy efficiency of the suggested paths.**

***Index Terms*—Combinatorial optimization learning, Dijkstra's algorithm, energy-efficiency, UAV, wireless backhauling**

I. Introduction

As an emerging technology, beyond 5G (B5G)/6G wireless networks will support the integration of wireless intelligence that is well tailored for resource management, edge/cloud computing, and robotics [1]–[3]. However, there still exists the fundamental issue of limited wireless network capacity as the traffic demands from ground users (GUs) grow exponentially due to the proliferation of connected devices. This motivates the ultra densification of small cells to increase the overall network capacity and meet users' traffic requirements in the B5G/6G networks. The traditional method is to install fixed small-cell base stations (FBSs) with millimeter-wave (mmWave) wireless backhauling to the macro-BS to cover the whole network coverage area [4]–[6]. While effective, this approach significantly impacts capital expenditures due to the installation and maintenance of FBSs [7], [8]. Moreover, it lacks flexibility in accommodating dynamic flow demands, which can vary across locations and time [9].

Unmanned aerial vehicle (UAV) base stations (UBSs) are becoming a part of the B5G/6G wireless network infrastructure [9], [10], and serve as relay nodes for mmWave wireless backhauling [9]–[12], leveraging their ability to grasp tall buildings and lampposts. UBSs are better than FBSs due to their on-demand deployment by providing enhanced network capacity and improved connectivity. In addition, UBSs are designed to operate with low transmission power, allowing for prolonged service time using robotic end effectors and dexterous grippers to grasp lampposts [13], unlike traditional FBSs which always incur huge power consumption. According to [9], [13], UBSs reduce the enormous energy consumption required for tasks such as flying and hovering by harvesting solar energy [14] or applying a free space optics-based charging system [15].

Some important considerations in the deployment of UBSs include achieving robust backhaul connectivity, ensuring seamless data transfer between GUs and macro-BS, and minimizing the total energy consumption. Recent advances such as the prioritization of certain links [16] and artificial intelligence-enabled decision making [17] have made UAV-assisted wireless backhauling promising. However, this comes with a limitation on the ability to respond quickly to dynamic scenarios. For instance, when there is a change in the source-destination pair of commodity flows or when there is a change in the required throughput of similar flow demands, the fixed path becomes inefficient, as some links may not satisfy all route constraints. Likewise, a fixed path will prevent the reachability of the UBSs to the macro-BS when natural disasters such as a hurricane or earthquake occurs, as those links automatically become inexistent or useless due to destroyed lampposts or buildings. In addition, since the communication range of UBSs is short, prioritizing one link will compromise the usability of other links in the network. For these reasons, UAV-assisted wireless backhauling deserves our attention. Specifically, the fundamental challenge lies in optimizing the number of UBSs to be deployed, their locations, and the multi-commodity flow paths, while satisfying certain user traffic

This work was supported in part by the NSF under grant CNS-2008092.

O. T. Ajayi and Y. Cheng are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, 60616. (e-mail: oajayi6@hawk.illinoistech.edu, cheng@illinoistech.edu)

S. Wang is with the School of Computer Science and Engineering, California State University, San Bernardino, San Bernardino, CA, 92407. (email: suyang.wang@csusb.edu)

requirements, routing constraints, and energy consumption constraints.

The routing problem can be formulated as a *multi-hop multi-commodity flow optimization* problem, which is a mixed integer linear programming (MILP) problem with NP-hardness. Some works [9], [18] have proposed mathematical and heuristic algorithms to approximate the solution to the multi-hop multi-commodity flow optimization problem. The work in [9] presented the UAV-assisted wireless backhauling as a multi-commodity low-cost flow problem and solved it using an optimization solver in MATLAB. Likewise, [18] proposed a greedy algorithm to reduce the search space for the solution to the routing problem. However, the computational overhead incurred by the mathematical and greedy algorithms is a matter of concern in 6G wireless networks where ultra-reliable low latency communication is one of the key attributes. In addition, running the optimization solver every time for the MILP problem in large-scale networks to compute the best paths is inefficient even if the network topology remains fixed.

As machine learning (ML) continues to revolutionize many fields, its application to wireless network optimization research is becoming popular. With ML, an optimization problem is treated as a mapping function, the output being the optimization solution given the input of the problem [19]–[22]. In many studies, ML has been leveraged to tackle the computational cost that renders conventional optimization algorithms ineffective. The works in [23]–[25] focused on the transmission scheduling problem in wireless networks. In the studies, the proposed ML approach aimed at learning the scheduling classes given the problem inputs, such that the commodity flows are forwarded over multiple hops based on the scheduling information obtained through ML. Since the key issue in backhauling is routing, the ML approaches in [23]–[25] are not applicable, as they only addressed the transmission scheduling and flow forwarding problems.

As an intersection between ML and mathematical optimization, combinatorial optimization learning is increasingly gaining attention in numerous fields. This is because of its unique advantages which include learning to optimize without explicit models by leveraging patterns from data, ability to solve hard combinatorial optimization instances with reduced computational overhead, generalization to unseen problem instances, and support for online and context-aware decision making with high accuracy. In this paper, we demonstrate the paradigm of "combinatorial optimization learning" for tackling the routing problem in wireless backhauling. Specifically, we examine the energy-efficient routing problem in a UAV-assisted wireless backhauling network similar to that studied in [9]. For the multi-hop multi-commodity flow optimization problem, we can interpret the computation of the routing path of each commodity flow as a mapping function, where the input features are the network topology, the source/destination of the tagged commodity, and the deployment of other commodities as resource competitors, while the output is the best path for the tagged commodity flow. We develop a combinatorial optimization learning approach (COLA) to implement this idea. The COLA consists of a two-stage design. The first stage is a neural network that can predict the "magic" link weights given the application-level inputs. Such link weights are fed into the second stage to compute the path for a target commodity flow by the simple Dijkstra's algorithm. Note that the proposed COLA can compute the energy-efficient path for each commodity flow with very low complexity: the ML layers (which could be as simple as just one layer as shown by our experiments) plus the simple Dijkstra's shortest-path algorithm.

Nevertheless, the proposed COLA in this paper is by no means trivial considering the following: 1) the link weights need to be intelligently computed by the neural network in the first stage, in order to shape the suggested path using Dijkstra's algorithm, to be as close as the best path for the original optimization problem, and 2) the gradient backpropagation requires a careful design to handle the differentiability over the combinatorial component and ensure model convergence. In this paper, we will demonstrate how to decompose the original multi-hop multi-commodity flow optimization problem into learnable single-commodity shortest path problems, how to obtain the training data, and how to train the system handling both the differentiability and convergence issues.

We evaluate our proposed COLA on small-sized and large-sized wireless networks showing significant benefits in computation time reduction, while maintaining near-optimal solution in terms of UBSs deployment accuracy and path energy efficiency. This paper further provides results with insights to assist researchers in understanding the effects of neural network design choice, loss function selection, and hyperparameter configurations when implementing COLA.

Our main contributions in this paper can be summarized as follows:

- The proposed COLA framework explores the paradigm of combinatorial optimization learning, and, for the first time, leads to an efficient ML-assisted approach that can timely and dynamically compute the end-to-end paths for optimal traffic deployment in B5G/6G wireless backhaul networks.
- We design a "data decomposition" technique for the multi-commodity flows to facilitate COLA training and can be seamlessly applied during inference.
- We provide detailed design and implementation of COLA to approximate the solution to the multi-commodity flow optimization problem. This includes gradients backpropagation incorporating the combinatorial component, and the adaptive perturbation of link weights to facilitate training convergence.
- Our results offer clear insights into how and why COLA outperforms other ML techniques, thus making COLA a viable approach for solving complex wireless network optimization problems.

The remainder of this paper is organized as follows. We present the system model and the problem formulation in Section II, followed by the COLA system design in Section III. Next, we present the proposed COLA for energy-efficient wireless backhauling in Section IV, followed by numerical experiments and results in Section V. We give some related work in Section VI and conclude the paper in Section VII. The list of acronyms used in this paper is summarized in Table I.

TABLE I: List of acronyms.

Acronym	Meaning
COLA	Combinatorial Optimization Learning Approach
CTR	Computation Time Reduction
DQN	Deep Q-Network
EER	Energy Efficiency Ratio
FBS	Fixed Base Station
GU	Ground User
HA	Hotspot Area
ILP	Integer Linear Programming
LoS	Line of Sight
MILP	Mixed Integer Linear Programming
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
mmWave	Millimeter Wave
PMA	Path Matching Accuracy
RL	Reinforcement Learning
SLP	Single Layer Perceptron
UAV	Unmanned Aerial Vehicle
UBS	UAV Base Station
UDA	UBS Deployment Accuracy

II. System Model and Problem Formulation

We consider an urban geographical area that has a defined layout of streets with high-rise buildings and lampposts that are suitable for the deployment of UBSs to backhaul the traffic of GUs in a B5G/6G mmWave network. The UBSs are equipped with robotic end effectors to grasp lampposts, thus alleviating energy consumption for flying and/or hovering. At any given time, increased traffic demand is observed in dense GU locations (i.e., hotspot areas (HAs)), and all GUs' traffic is to be backhauled to the macro-BS via mmWave links. We represent the wireless network as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes comprising: 1) the set of lampposts with a macro-BS denoted by $\mathcal{V}_L = \{1, \cdots, L, 0\}$ (with 0 denoting the macro-BS and L denoting the number of lampposts), and 2) the set of UBSs denoted by $\mathcal{U} = \{1, \cdots, U\}$, where $|\mathcal{U}| \leq L$, which means that the number of UBSs should not exceed the number of possible deployment locations. We denote the set of HAs by $\mathcal{V}_H = \{1, \cdots, H\}$ with flow demands $\mathcal{D} = \{1, \cdots, D\}$, where $d \in \mathcal{D}$ is the source-destination tuple of a flow demand, $|\mathcal{V}_H| = |\mathcal{D}|$ is the number of HAs, and th_d is the amount of GUs' traffic to be backhauled for flow d. Note that each HA has a lamppost that a UBS can grasp for wireless backhauling. The set of edges/links is denoted by $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}_L, i \neq j\}$. The macro-BS coordinates the deployment of UBSs within the network. A deployed UBS provides wireless access to its GUs or serves as a relay node to the macro-BS or both. For simplicity, we use "nodes" as a generic term to refer to lampposts and/or UBSs throughout this paper.

Since our setting considers the mmWave frequency band, we assume a line of sight (LoS) condition for the backhaul links. This assumption is justified by the deployment of UBSs atop tall buildings and lampposts in the urban area, where objects can rarely obstruct the links. Moreover, mmWave communication employs directional beamforming with high-gain antennas for signal propagation to reduce the effect of channel loss [5], [18]. Consequently, mmWave signals become less susceptible to neighborhood interference from other links. As such, we only consider the signal-to-noise ratio when computing the wireless backhaul capacity c_{ij} of link (i, j) as follows:

$$c_{ij} = B \min \left[S_{max}^{eff}, \log_2 \left(1 + \frac{P_i^{tx} d_{ij}^{-\alpha} G_{ij}}{\sigma B} \right) \right] \tag{1}$$

where P_i^{tx}, d_{ij}, α, G_{ij}, σ, B, and S_{max}^{eff} denote the unit transmit power of node i, the Euclidean distance between two nodes, the distinct LoS path loss exponent, the gain of the directional antenna, the noise power at the receiver node, the channel bandwidth, and the maximum spectral efficiency in bits per second per hertz, respectively. To avoid signal quality degradation, backhaul links are established between candidate locations that are within a specified range (usually short range), as long-distance links will cause severe path loss.

Based on the system model defined for $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we formulate the multi-commodity flow optimization problem for the energy-efficient UAV-assisted wireless backhauling in B5G/6G networks. The problem formulation incorporates several constraints with an objective to minimize the number of UBSs to be deployed, the number of hops, and the total energy consumption as follows:

$$\min_{\{x_{iu}\}, \{y_{ijd}\}, \{e_{iu}^{total}\}} \sum_{i \in \mathcal{V}_L} \sum_{u \in \mathcal{U}} x_{iu} + \sum_{i \in \mathcal{V}_L} \sum_{j \in \mathcal{V}_L} \sum_{d \in \mathcal{D}} y_{ijd} + \sum_{i \in \mathcal{V}_L} \sum_{u \in \mathcal{U}} e_{iu}^{total} \tag{2}$$

$$\text{s.t.} \quad \sum_{u \in \mathcal{U}} x_{iu} \leq 1, \quad \forall i \in \mathcal{V}_L \setminus \{0\}, \tag{3a}$$

$$\sum_{i \setminus \{0\} \in \mathcal{V}_L} x_{iu} \leq 1, \quad \forall u \in \mathcal{U}, \tag{3b}$$

$$\sum_{j \in \mathcal{V}_L} y_{ijd} = 1, \quad \forall i \in \mathcal{V}_H, \quad \forall d \in \mathcal{D}, \tag{4a}$$

$$\sum_{i (i \neq 0) \in \mathcal{V}_L} y_{i0d} = 1, \quad \forall d \in \mathcal{D}, \tag{4b}$$

$$\sum_{j (j \neq i) \in \mathcal{V}_L} y_{ijd} = \sum_{j (j \neq i) \in \mathcal{V}_L} y_{jid}, \quad \forall i \in \mathcal{V}_L \setminus \{0\}, \quad \forall d \in \mathcal{D}, \tag{4c}$$

$$\sum_{u \in \mathcal{U}} x_{iu} \geq y_{ijd}, \quad \forall i \in \mathcal{V}_L \setminus \{0\}, \quad \forall j \in \mathcal{V}_L, \quad \forall d \in \mathcal{D}, \tag{4d}$$

$$\sum_{d \in \mathcal{D}} y_{ijd} th_d \leq c_{ij}, \quad \forall (i, j) \in \mathcal{E}, \tag{5}$$

$$e_{iu}^{total} = x_{iu} e_t^{0i} + x_{iu} e_g + \sum_{d \in \mathcal{D}} y_{ijd} e_c, \quad \forall i, j \in \mathcal{V}_L, \quad \forall u \in \mathcal{U}, \tag{6}$$

$$x_{iu} \in \{0, 1\}, \quad \forall i \in \mathcal{V}_L \setminus \{0\}, \quad \forall u \in \mathcal{U}, \tag{7}$$

$$y_{ijd} \in \{0, 1\}, \quad \forall i, j (i \neq j) \in \mathcal{V}_L, \quad \forall d \in \mathcal{D}, \tag{8}$$

$$e_{iu}^{total} \geq 0, \quad \forall i \in \mathcal{V}_L \setminus \{0\}, \quad \forall u \in \mathcal{U}, \tag{9}$$

The optimization problem in (2) is an MILP problem with an exponential solution space, and is thus NP-hard, with complexity in the order $\mathcal{O}(|\mathcal{D}|(2^{|\mathcal{E}|}) + 2|\mathcal{V}_L|^2)$. We define the binary decision variables in (7) and (8) as follows:

$$x_{iu} = \begin{cases} 1, & \text{if UBS } u \text{ is dispatched to lamppost } i \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ijd} = \begin{cases} 1, & \text{if link } (i,j) \text{ carries commodity flow } d \\ 0, & \text{otherwise} \end{cases}$$

Constraints (3a) and (3b) ensure that at most one UBS stays at node i. Constraints (4a)-(4d) enforce that a flow d begins from the hotspot area i and ends at the macro-BS, with flow conservation at intermediate nodes (i.e., the net flow at intermediate nodes is zero) and multi-flow traversing a link. We assume that there are sufficient radio frequency channel resources to support multiple flows on a link. Constraint (5) ensures that the sum of all GUs' traffic th_d that needs to be backhauled to the macro-BS for all flows traversing link (i,j) does not exceed its capacity. Note that without constraints (6) and (9), the optimization problem would be reduced to an integer linear programming (ILP) problem which is easier to solve. The objective function of the ILP problem would minimize the number of hops and the number of UBSs to be deployed. However, we include the energy constraint in (6) to ensure that UBSs operate with minimal energy consumption, and define the continuous variable for total energy consumption in (9). The total energy consumption e_{iu}^{total} of UBS u when operating at node i is the sum of the traveling energy e_t^{0i} from the macro-BS to node i, the grasping energy e_g using its grippers [26], and the communication energy e_c for all the flows. Eqs. (10)-(12) mathematically define these components:

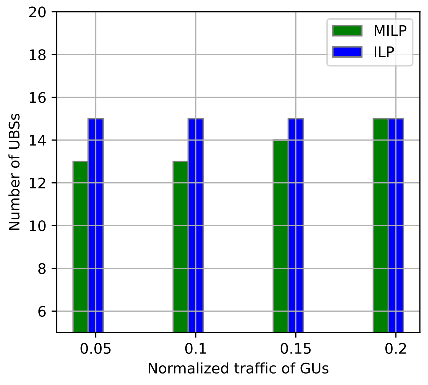
$$e_t^{0i} = P_n \frac{d_{0i}}{v} \tag{10}$$

$$e_g = P_g t \tag{11}$$

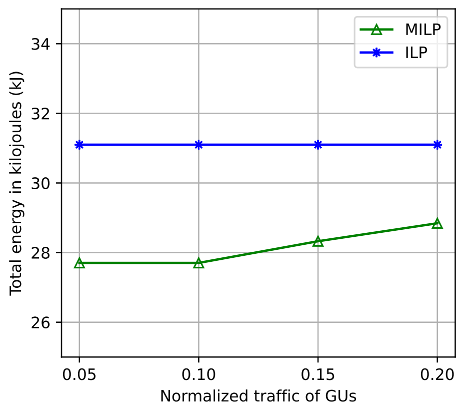
$$e_c = (P_a + \eta_p P_i^{tx})t \tag{12}$$

where P_n is the propulsion power, d_{0i} is the Euclidean distance between the macro-BS and node i, and v is the velocity of the UBSs in meter per second (m/s). For the grasping energy, P_g is the grasping power, and t is the time in seconds, while for the communication energy, P_a denotes the minimum active power of the UBS, and η_p is a linear transmission factor [27].

We compare the solutions to the ILP problem and the MILP problem w.r.t. the number of UBSs and the total energy consumption in Fig. 1. When the normalized traffic of GUs for each HA is 0.1, the optimal number of UBSs is 13 for the MILP problem and 15 for the ILP problem, as shown in Fig. 1a. The optimal number of UBSs for the ILP problem is constant for all the normalized GUs' traffic. This consistency suggests that the ILP problem models a shortest-path problem where the costs of all links are uniform. In contrast, the MILP problem is considered to be a shortest-path problem with link costs being the energy consumption of the UBSs. Hence, the MILP problem is more relevant to the UAV-assisted wireless backhauling in B5G/6G multi-hop networks where an increase in GUs' traffic incurs more UBSs deployment, and as a consequence, increased energy costs, as shown in Fig. 1a and Fig. 1b, respectively.



(a) Optimal number of UBSs.



(b) Energy consumption.

Fig. 1: Comparing MILP and ILP. Illustration is based on a 5×5 topology and $|\mathcal{V}_H| = 10$.

III. COLA System Design

The concept of combinatorial optimization learning is introduced as a method that leverages the output of neural networks to solve combinatorial optimization problems that have linear objective functions and finite discrete state spaces [28], [29]. In this section, we first present the shortest-path computation for a single-commodity flow, and then describe the methodology of gradients backpropagation over the combinatorial component in our proposed COLA.

A. Shortest-Path Computation

The optimization problem defined in (2)-(9) can be modeled as a mapping function with supervised learning. To solve the multi-commodity flow optimization problem, we first decompose it into single-commodity flow shortest-path problems and design COLA to quickly and effectively solve it.

For a single-commodity flow, we can compute the shortest path using a simple low-complexity Dijkstra's algorithm in COLA. Specifically, as shown in Fig. 2, the first stage in COLA consists of a neural network that takes the application-level information A_m^d as input, and predicts the link weights $\hat{W}_m^d$ to be fed into the Dijkstra's shortest-path algorithm. In the second stage, given the link weights $\hat{W}_m^d$, the Dijkstra's algorithm simply finds the shortest path from the source of the

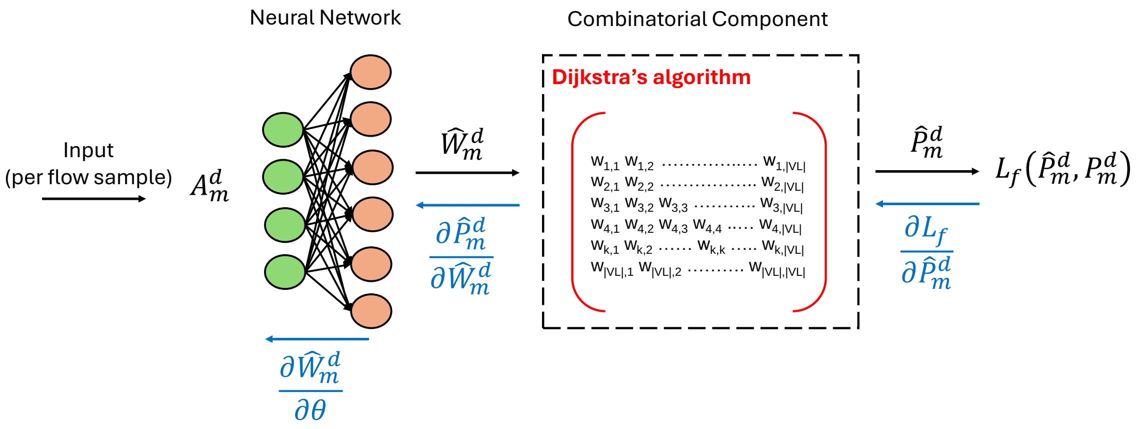

Neural Network
Combinatorial Component
Dijkstra's algorithm
Input (per flow sample)
A_m^d
$\hat{W}_m^d$
$\hat{P}_m^d$
$L_f(\hat{P}_m^d, P_m^d)$
$\frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d}$
$\frac{\partial L_f}{\partial \hat{P}_m^d}$
$\frac{\partial \hat{W}_m^d}{\partial \theta}$
$w_{1,1}\ w_{1,2}\ \dots\ w_{1,|VL|}$
$w_{2,1}\ w_{2,2}\ \dots\ w_{2,|VL|}$
$w_{3,1}\ w_{3,2}\ w_{3,3}\ \dots\ w_{3,|VL|}$
$w_{4,1}\ w_{4,2}\ w_{4,3}\ w_{4,4}\ \dots\ w_{4,|VL|}$
$w_{k,1}\ w_{k,2}\ \dots\ w_{k,k}\ \dots\ w_{k,|VL|}$
$w_{|VL|,1}\ w_{|VL|,2}\ \dots\ w_{|VL|,|VL|}$

Fig. 2: Proposed COLA framework for UAV-assisted wireless backhauling.

single-commodity flow to the destination. The path suggested by the Dijkstra's algorithm is denoted by $\hat{P}_m^d$, which can be expressed as:

$$\hat{P}_m^d = \text{Dijkstra}(\hat{W}_m^d) \tag{13}$$

where $\hat{W}_m^d = f_\theta(A_m^d)$ is the output of the neural network (denoted by f_θ). With proper data design, COLA can be implemented to compute the shortest paths for the multi-commodity flow.

TABLE II: List of notations.

Notation	Description
$\mathcal{G}(\mathcal{N},\mathcal{E})$	Network graph with node set $\mathcal{N}$ and edge set $\mathcal{E}$
$\mathcal{V}_L$	Set of lampposts and macro-BS
$\mathcal{V}_H$	Set of HAs
$\mathcal{U}$	Set of UBSs
$\lvert . \rvert$	Number of elements in a set
S_{max}^{eff}	Maximum spectral efficiency
c_{ij}	Backhaul capacity on link (i, j)
e_{iu}^{total}	Total energy consumption by UBS u at node i
λ	Perturbation hyperparameter for link weights
ϵ_λ	Control parameter for λ
$U_{\text{OPT}}^{d,m}$	Optimal set of UBSs for flow d of instance m
$U_{\text{COLA}}^{d,m}$	COLA's set of UBSs for flow d of instance m
$\hat{W}_m^d$	Predicted link weights
$W_m^{'d}$	Perturbed link weights
P_m^d	Ground truth path
$\hat{P}_m^d$	Suggested path based on $\hat{W}_m^d$
$\hat{P}_{\lambda m}^d$	Suggested path based on $W_m^{'d}$

B. Backpropagation of Gradients

To train the COLA system in Fig. 2, the gradient computation needs to be properly defined because of the combinatorial component. A key challenge arises due to the discrete nature of the combinatorial algorithm's output, which provides zero gradients and renders backpropagation impossible. To tackle the differentiability challenge and update the neural network parameters θ, a hyperparameter $\lambda > 0$ is introduced. This perturbs the predicted link weights[1] $\hat{W}_m^d$, yielding meaningful gradients for backpropagation [29]. The gradient of the loss w.r.t. the neural network parameters is defined as:

$$\frac{\partial L_f}{\partial \theta} = \frac{\partial L_f}{\partial \hat{P}_m^d} \cdot \frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d} \cdot \frac{\partial \hat{W}_m^d}{\partial \theta} \tag{14}$$

where L_f is the loss function that measures the hamming distance or mean squared error between the ground truth path P_m^d and the suggested path $\hat{P}_m^d$. In eq. (14), both $\frac{\partial L_f}{\partial \hat{P}_m^d}$ and $\frac{\partial \hat{W}_m^d}{\partial \theta}$ can be easily computed for backpropagation, but it is impossible to directly compute $\frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d}$.

To find $\frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d}$, a function $f_\lambda(\hat{W}_m^d)$ is introduced as a continuous interpolation of $\hat{P}_m^d$ whose gradient computation is defined as:

$$\nabla f_\lambda(\hat{W}_m^d) = -\frac{1}{\lambda}[\hat{P}_m^d - \hat{P}_{\lambda m}^d] \tag{15}$$

where $\hat{P}_{\lambda m}^d$ is the output of the Dijkstra's algorithm using perturbed link weights $W_m^{'d}$ as follows:

$$W_m^{'d} = \hat{W}_m^d + \lambda \cdot \frac{\partial L_f}{\partial \hat{P}_m^d} \tag{16}$$

$$\hat{P}_{\lambda m}^d = \text{Dijkstra}(W_m^{'d}) \tag{17}$$

Therefore, $\nabla f_\lambda(\hat{W}_m^d)$ is returned as a reasonable replacement for $\frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d}$ in eq. (14). Apparently, the Dijkstra's algorithm is used in the forward propagation to obtain $\hat{P}_m^d$ and in the backpropagation to obtain $\hat{P}_{\lambda m}^d$. We present the COLA training procedure in Algorithm 1. Lines 1 to 4 compute the forward pass by predicting the link weights and using Dijkstra's algorithm to obtain the shortest path. Lines 5 to 10 perturb the link weights for backpropagation in updating the parameters of the neural network f_θ.

IV. COLA for UAV-Assisted Wireless Backhauling

In this section, we present the implementation of COLA for the multi-hop multi-commodity flow optimization problem. This includes the design of the training data, the COLA framework, and the methodology for achieving training convergence. The list of notations used in this paper is summarized in Table II.

[1]The hyperparameter λ should be chosen to cause a noticeable change in the solution to the optimization problem, otherwise gradients will be zero.

Algorithm 1 COLA Training Algorithm

```
Require: Initialize the parameters of $f_\theta$ and pre-define $\lambda$
                                                        ▷ FORWARD PASS.
 1: Obtain the predicted link weights $\hat{W}_m^d = f_\theta(A_m^d)$
 2: Solve the shortest path $\hat{P}_m^d$ = Dijkstra($\hat{W}_m^d$) using the predicted link weights
 3: save $\hat{P}_m^d$ and $\hat{W}_m^d$ for backpropagation
 4: return the suggested path $\hat{P}_m^d$
                                                        ▷ BACKPROPAGATION.
 5: load $\hat{P}_m^d$ and $\hat{W}_m^d$ from the forward pass
 6: Compute perturbed link weights $W_m^{'d}$ according to eq. (16)
 7: Solve $\hat{P}_{\lambda m}^d$ using $W_m^{'d}$ as in eq. (17)
 8: Compute gradient $\nabla f_\lambda(\hat{W}_m^d)$ as in eq. (15)
 9: $\frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d} \leftarrow \nabla f_\lambda(\hat{W}_m^d)$
10: return $\frac{\partial \hat{P}_m^d}{\partial \hat{W}_m^d}$ to compute eq. (14)
11: update the neural network parameters $\theta$
```

A. Decomposition of the Training Data

In this paper, we design the training data for COLA by generating problem instances for the MILP problem defined in (2)-(9) and solving them with an optimization solver (e.g., Gurobi) to obtain the optimal solutions w.r.t. the UBSs deployed and energy-efficient paths. The dataset denoted by $\mathcal{D}_S = \{S_1, \cdots, S_M\}$ contains M instances, where each S_m represents a multi-commodity flow problem instance. It is worthy to note that the optimal true paths of the multi-commodity flow are not necessarily a shortest-path tree. This is because the link capacity constraint in (5) can cause the incoming flows to a node to traverse different outgoing links. This behavior aligns with the multiple-source to single-destination macro-BS setting described in Section II. However, our aim is to solve the routing problem for the multi-commodity flows with a low-complexity algorithm. We leverage the well-known Dijkstra's algorithm [30] to achieve this aim.

First, we develop a *data decomposition technique* where a problem instance is decomposed into single-commodity flow samples with each sample treated as being distinct. For a problem instance S_m that has D_m flow demands, the samples are $S_m = \{(A_m^d, P_m^d)_{d=1}^{D_m}\}$, where A_m^d is the application-level input for the d-th flow demand in S_m and P_m^d is the corresponding ground truth path. Hence, the training data is the collection of all the samples of the problem instances denoted as $\mathcal{D}_\mathcal{S} = \{(A_1^d, P_1^d)_{d=1}^{D_1}, \cdots, (A_M^d, P_M^d)_{d=1}^{D_M}\}$, and the size of the dataset is $|\mathcal{D}_S| = \sum_{m=1}^{M} D_m$. This means that with the "data decomposition" technique, the training data contains single-commodity flow samples which can be solved using Dijkstra's shortest-path algorithm in COLA.

The application-level input A_m^d is a vector which contains the network topology information N_t and the normalized traffic of GUs th_d for each flow in S_m, which is denoted by T_d, where $T_d = (th_1, \cdots, th_{D_m})$. The vector N_t is a $|\mathcal{V}_L|$-dimensional vector, where $|\mathcal{V}_L|$ corresponds to the number of lampposts and the macro-BS in the network. The elements of the vector N_t are represented as follows:

$$N_t(a) = \begin{cases} 1, \text{ if node } a \text{ is the source of flow } d \\ 0.5, \text{ if node } a \text{ is a source of other flows in } S_m \\ 2, \text{ if node } a \text{ is the macro-BS} \\ 0, \text{ otherwise} \end{cases}$$

where $N_t(a)$ is the value at position a in vector N_t, and $N_t(a) = 0.5$ models the deployment of other commodities as resource competitors. The vector T_d has its dimension being fixed to the maximum possible number of flow demands the network can handle per time. Hence, the application-level input vector A_m^d has a fixed dimension as follows:

$$A_m^d = Concat\,(N_t, T_d) \tag{18}$$

where $Concat$ is the concatenation operation for vectors.

The ground truth path for flow d is represented as an adjacency matrix $P_m^d \in \{0,1\}^{|\mathcal{V}_L| \times |\mathcal{V}_L|}$ as follows:

$$P_m^d[i,j] = \begin{cases} 1, \text{ if link } (i,j) \in \mathcal{E} \text{ is used for flow } d \\ 0, \text{ otherwise (including diagonal elements)} \end{cases}$$

where $P_m^d[i,j]$ is the value at row i and column j of the adjacency matrix. We also denote the set of the optimal UBSs by $U_{\text{OPT}}^{d,m} = \text{unique}\{i, j \mid P_m^d[i,j] = 1, \quad \forall (i,j) \in \mathcal{E}\}$ which contains the unique indices of the UBSs in the ground truth path P_m^d.

B. COLA Framework

The COLA framework shown in Fig. 2 consists of a neural network and Dijkstra's algorithm. The neural network takes the application-level information of each flow sample as input and outputs a link weights vector $\hat{W}_m^d$, which is structured as a matrix. The matrix $\hat{W}_m^d$ is then fed into Dijkstra's algorithm which computes the shortest path based on the link weights. We denote the set of the UBSs suggested by COLA as $U_{\text{COLA}}^{d,m} = \text{unique}\{i, j \mid \hat{P}_m^d[i,j] = 1, \quad \forall (i,j) \in \mathcal{E}\}$ which contains the unique indices of the UBSs in $\hat{P}_m^d$, where $\hat{P}_m^d$ is the shortest path suggested by COLA.

1) Neural Network: The neural network in the COLA framework is a single-layer perceptron (SLP), which contains an input layer and one output layer with linear activation, where the number of neurons in the output layer corresponds to the size of the shortest path $\hat{P}_m^d$ matrix. The predicted link weights can be obtained as follows by the SLP:

$$\hat{W}_m^d = \theta A_m^d + b \tag{19}$$

where $\theta \in \mathbb{R}^{|\mathcal{V}_L|^2 \times dim(A_m^d)}$ is the parameter matrix, $\hat{W}_m^d \in \mathbb{R}^{|\mathcal{V}_L|^2}$ is the link weights to be used by the Dijkstra's algorithm, $b \in \mathbb{R}^{|\mathcal{V}_L|^2}$ is the bias vector, and A_m^d is a column vector (the $dim(.)$ denotes the dimension of a vector).

We opine that the SLP mitigates the problem of vanishing gradients which deep neural networks suffer from. The gradients incorporating the combinatorial component should directly adjust the parameters θ of the neural network, which in turn directly updates the link weights $\hat{W}_m^d$ in the forward propagation. In the experiments and results in Section V, we compare the performance of the SLP with that of a multi-layer perceptron (MLP) to evaluate its efficiency and effectiveness.

2) Dijkstra's Algorithm for Approximate Solutions: The Dijkstra's algorithm is the combinatorial algorithm used by COLA to find the approximate solution to the original MILP problem. It is a viable algorithm for finding the shortest path from a source node to all other nodes in the network, and also offers low-time complexity in the order $\mathcal{O}(|\mathcal{V}_L|^2)$. The link weights vector $\hat{W}_m^d$ obtained from the neural network is transformed into a matrix using a *Vector-to-Matrix* operation and is used by Dijkstra's algorithm to compute the shortest path from the source of the flow demand to the macro-BS. The link weight matrix is as follows:

$$\hat{W}_m^d[i,j] = \begin{cases} w_{i,j}, & \text{if link } (i,j) \in \mathcal{E} \\ \infty, & \text{otherwise (including diagonal elements)} \end{cases}$$

where $w_{i,j}$ is the value at the i-th row and j-th column of the $\hat{W}_m^d$ matrix, serving as the weight for link $(i,j) \in \mathcal{E}$. We set the value of $(i,j) \notin \mathcal{E}$ to infinity to represent an invalid or non-existent link, ensuring that Dijkstra's algorithm avoids using these links in routing.

To show that COLA provides a feasible solution to the optimization problem, we prove in Proposition 1 that $w_{i,j}$ is a surrogate for the energy constraint e_j^{total} in (6) and that the shortest paths suggested by the Dijkstra's algorithm are the energy-efficient paths.

Proposition 1. *Assume that flow d arrives at UBS i (meaning a UBS is in node i), and the set of outgoing links from the UBS is $\{(i,j),(i,k)\}$. By Dijkstra's algorithm, flow d will only traverse link (i,j) if $w_{i,j} < w_{i,k}$ which indicates that link (i,j) has a lower cost compared to (i,k).*

Proof. Let us define $w_{i,j}$ and $w_{i,k}$ as follows:

$$w_{i,j} = e_i^{total} + e_j^{total} \tag{20}$$

$$w_{i,k} = e_i^{total} + e_k^{total} \tag{21}$$

Subtracting eq. (20) from eq. (21) yields:

$$w_{i,k} - w_{i,j} = e_k^{total} - e_j^{total} \tag{22}$$

If $e_k^{total} - e_j^{total} > 0$, then $w_{i,k} > w_{i,j}$. For link (i,j) to be in the path, it means $w_{i,j} < w_{i,k}$, which implies that $e_j^{total} < e_k^{total}$. Hence, link (i,j) is used for flow d. The link capacity constraint (5) is also satisfied, i.e., $y_{ijd}th_d \leq c_{ij}$. □

From the proof, we conclude that a UBS will be deployed to node j rather than node k. Next, we prove in Proposition 2 that Dijkstra's algorithm can find the shortest path that is a feasible solution for each commodity flow in the MILP problem.

Proposition 2. *Consider a flow demand at a given source node. A UBS will be deployed to this node to begin the backhauling of the GUs traffic. This satisfies constraint (4a). Dijkstra's algorithm will then find the next unvisited node with the least energy cost to the destination node.*

Proof. Let s denote the source node that generates traffic to be backhauled to the macro-BS. Let $\mathcal{K} = \{1, \cdots, K\} \setminus \{s\}$ and $C = \{c_1, \cdots, c_K\}$ denote the set of unvisited nodes in the network and their associated costs, respectively. The cost c_k is the total energy cost in getting to the destination macro-BS through node k. Hence, Dijkstra's algorithm selects the next node k according to:

$$k = \arg\min_{k \in \mathcal{K}} c_k \tag{23}$$

and a UBS is dispatched to that node. This continues until all the nodes in the network, including the macro-BS, are reachable from source node s. This satisfies constraint (4b). □

The integration of Dijkstra's algorithm in the COLA framework helps to find a feasible solution which satisfies the constraints of the MILP problem. At inference time, for a given problem instance S_m with multi-commodity flows, COLA is executed for D_m rounds to obtain the corresponding paths.

3) Loss Function: We use the mean squared error (MSE) loss function for COLA. Unlike the Hamming loss, which is non-differentiable, MSE allows gradient computation for training. The MSE loss function L_f is defined as:

$$L_f = \frac{1}{|\mathcal{V}_L|^2} \sum_{i=1}^{|\mathcal{V}_L|} \sum_{j=1}^{|\mathcal{V}_L|} (P_m^d[i,j] - \hat{P}_m^d[i,j])^2 \tag{24}$$

C. Link Weights Perturbation for COLA Convergence

The backpropagation of the gradients to update the neural network parameters θ requires choosing an appropriate value for the perturbation hyperparameter $\lambda > 0$. A very small λ causes the model to get stuck in a saddle point, resulting in zero gradients after a few epochs. On the other hand, a very large λ causes severe divergence, where the perturbed link weights $W_m^{'d}$ deviate from the optimal weights.

To tackle this problem, we propose a method for selecting an appropriate λ value in Proposition 3. We also introduce a parameter ϵ_λ to adaptively control the adjustment of λ during training towards COLA convergence.

Proposition 3. *To obtain the perturbed link weights $W_m^{'d} \neq \hat{W}_m^d$, the value of λ is important, as it determines whether or not COLA can suggest good paths for a problem instance.*

Proof. From eq. (24) we obtain $\frac{\partial L_f}{\partial \hat{P}_m^d} = \frac{2(\hat{P}_m^d - P_m^d)}{|\mathcal{V}_L|^2}$, which is the derivative of the loss w.r.t. the suggested path $\hat{P}_m^d$. Then, similar to eq. (16) we have:

$$W_m^{'d} = \hat{W}_m^d + \lambda \cdot \frac{2(\hat{P}_m^d - P_m^d)}{|\mathcal{V}_L|^2} \tag{25}$$

Rearranging eq. (25), we have:

$$\lambda = (W_m^{'d} - \hat{W}_m^d) \cdot \frac{|\mathcal{V}_L|^2}{2} \tag{26}$$

To find λ in eq. (26), we ignore the $(\hat{P}_m^d - P_m^d)$ in eq. (25) as it only gives the direction of the gradient. We can deduce that $\frac{|\mathcal{V}_L|^2}{2} >> 0$ (which depends on the number of nodes in the network) and since the goal is to have a perturbation such that $|W_m^{'d} - \hat{W}_m^d| \in (0,1]$, then we must have $0 < \lambda \leq \frac{|\mathcal{V}_L|^2}{2}$. □

Setting a fixed λ value during training causes the gradient to be zero after only a few epochs, i.e., $W_m^{'d} = \hat{W}_m^d$. To prevent this issue, we adaptively control the adjustment of λ using a control parameter $\epsilon_\lambda > 0$ as follows:

$$\lambda_{t+1} = \begin{cases} \lambda_t + \epsilon_\lambda, & \text{if } Loss_t \geq \min(Loss_1, \cdots, Loss_{t-1}) \\ \lambda_t, & \text{otherwise} \end{cases}$$

At epoch $t+1$, the λ value is updated based on the training loss improvement. Training stops when the loss converges after several epochs. We show in Section V-E6 the performance of COLA for the fixed λ and adaptive λ settings.

V. Numerical Experiments and Results

We numerically simulate a wireless network for mmWave backhauling at 28 GHz for typical Manhattan-type [9] urban environments with street blocks (i.e., buildings). The entire layout is a 1 km by 1 km square area, suitable for a grid-type topology with lampposts uniformly positioned along the intersection of the streets. The distance between two lampposts d_{ij} is 50 m in our setting, and this is also the transmission range of the UBSs. Depending on the needs for ultra-densification of small cells in an urban area, networks of different sizes can be deployed dynamically.

Our experiments demonstrate COLA's performance for a small-sized deployment area of 250 m by 250 m which is a 5×5 grid topology and a large-sized deployment area of 400 m by 400 m which is an 8×8 grid topology as shown in Fig. 3a and 3b, respectively. Some of the key parameters used for the experiments are listed in Table III, while the hyperparameters used for COLA are listed in Table IV. The experiments are run on an 11th Gen Intel® Core™ i9-11900KF Linux machine (Ubuntu OS) with 3.50GHz × 16 processors, GeForce RTX 3090 graphics, and a 32 GB RAM.

TABLE III: List of simulation parameters.

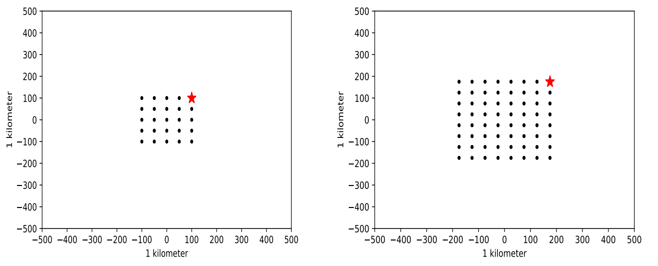
Parameter	Value
Bandwidth B	1 GHz
Normalized traffic of GUs th_d	(0,1] bps/Hz
LoS path loss exponent α	3.5
Maximum spectral efficiency S_{max}^{eff}	4.8 bps/Hz [5]
Transmit power p^{tx}	24 dBm [31]
Distance between two nodes d_{ij}	50 m
Noise power σ	-170 dBm
Propulsion power P_n	125 W [32]
Minimum active power P_a	6.8 W [27]
Linear transmission factor η_p	4 W [27]
Velocity v	10.21 m/s [32]
Time t	1 second

TABLE IV: List of COLA hyperparameters.

Parameter	Small-sized network	Large-sized network
Number of SLP neurons	$25^2 = 625$	$64^2 = 4096$
Perturbation parameter λ	250	1640
Control parameter ϵ_λ	50	330
Learning rate α	$5e^{-4}$	$5e^{-4}$
Batch size	300	400
Epochs	9	47
Regularization	early stopping	early stopping
Optimizer	Adam	Adam

A. Evaluation Metrics

The evaluation of COLA is based on three metrics defined as follows:



(a) Deployment area of the small-sized network (5×5) topology. (b) Deployment area of the large-sized network (8×8) topology.

Fig. 3: Illustration of grid-type layouts showing location of lampposts and macro-BS (the red top-right corner node).

1) UBSs Deployment Accuracy: The UBSs deployment accuracy (UDA) measures the similarity between the set of UBSs suggested by COLA and the set of optimal UBSs for a problem instance S_m. It is mathematically expressed as:

$$\text{UDA} = \left(\frac{\sum_{d=1}^{D_m} |U_{\text{OPT}}^{d,m} \cap U_{\text{COLA}}^{d,m}|}{\sum_{d=1}^{D_m} |U_{\text{OPT}}^{d,m} \cup U_{\text{COLA}}^{d,m}|} \right) \times 100 \tag{27}$$

The UDA metric is the higher the better and it is in percentage. A lower numerator means wrong UBSs selections and a higher denominator means dissimilarities between the two sets.

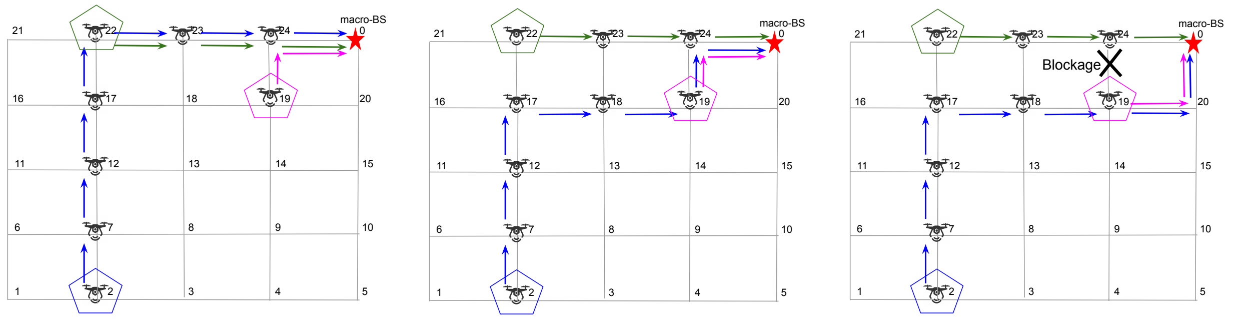
2) Energy Efficiency Ratio: This is the metric that describes the energy efficiency of the suggested paths by COLA. It is the ratio of the optimal total energy cost obtained by the benchmark solver (i.e., Gurobi) to the total energy cost obtained by COLA. Network service providers rely on this metric to analyze and estimate the operational cost when backhauling GUs' traffic to the macro-BS. The energy efficiency ratio (EER) for a problem instance S_m is computed as follows:

$$\text{EER} = \left(\frac{\sum_{d=1}^{D_m} \sum_{k \in U_{\text{OPT}}^{d,m}} e_k^{total}}{\sum_{d=1}^{D_m} \sum_{k \in U_{\text{COLA}}^{d,m}} e_k^{total}} \right) \times 100 \tag{28}$$

The EER metric is the higher the better and it is in percentage. The EER is low when the denominator increases, which indicates that the suggested paths deviate from the optimal energy-efficient paths by some degree. We opine that the EER metric is sufficient to measure COLA's performance in finding the best paths for a given optimization instance. This is because the solution to the MILP problem are often non-unique (i.e., there are multiple paths with the same total energy costs). Hence, for this reason, we do not explicitly evaluate the "path matching accuracy" (PMA) metric. Although in the ablation study of COLA, we use the PMA alongside the UDA to compare performance.

3) Computation Time Reduction: This measures the computation time reduction (CTR) when COLA is used to find the approximate solution for a problem instance relative to the computation time obtained from using an optimization solver like Gurobi to solve the original MILP problem. The CTR is computed as:

$$\text{CTR} = \left(\frac{t_{\text{OPT}} - t_{\text{COLA}}}{t_{\text{OPT}}} \right) \times 100 \tag{29}$$



(a) Optimal paths. (b) Suggested paths in LoS scenario. (c) Suggested paths in non-LoS scenario.

Fig. 4: Comparison between the optimal paths obtained by Gurobi and the paths suggested by COLA.

where t_{OPT} is the time it takes to find the optimal solution to the MILP problem using Gurobi, and t_{COLA} is the total computation time using COLA in predicting the "magic" link weights and finding the shortest paths with Dijkstra's algorithm. Both t_{OPT} and t_{COLA} are in seconds. The CTR metric is the higher the better and it is in percentage.

For the performance evaluation, we conveniently use the number of HAs, rather than the number of flow demands for the wireless backhauling context, but they have the same meaning, i.e., $|\mathcal{V}_H| = |\mathcal{D}|$.

B. *Performance in a Small-sized Network*

The small-sized network is a 5×5 grid topology which consists of 24 lampposts uniformly positioned within the 250 m by 250 m layout and a single macro-BS located on the top-right corner in the layout as illustrated in Fig. 3a. The lampposts are the candidate locations where the UBSs can be deployed for wireless backhauling, and the maximum number of UBSs is 24, which is to support heavy traffic situations (i.e., the deployment of the UBSs to all the lampposts). We run a simulation to generate the problem instances and their solutions using Gurobi optimization solver. The solution from Gurobi serves as the benchmark for the proposed COLA. For the problem instances, the number of HAs varies and the HAs are random. The training data consists of 71,000 instances for each $|\mathcal{V}_H| \in \{4, \cdots, 8\}$ making a total of 355,000 instances. The test data consists of 1,000 different instances for each $|\mathcal{V}_H| \in \{2, \cdots, 10\}$. This means that we evaluated COLA on smaller and larger numbers of HAs that are not seen during training. In order to capture dynamic scenarios, the generated instances have different normalized traffic for GUs.

1) UDA and EER of COLA: In Table V, we demonstrate the performance of COLA on all the metrics, and the result is an average of 1,000 instances for each number of HAs. COLA achieves up to 92.57% for the UDA when the number of HAs is 4, and for the case where the number of HAs is 9, COLA can achieve up to 83.97% for the UDA. The corresponding EER of 97.45% and 91.36%, respectively, shows that COLA's shortest-path solution is near-optimal. This is also evident in the light traffic scenarios of 2 HAs and heavy traffic scenarios of 10 HAs, where COLA achieves up to 98.51% and 91.11% for the EER, respectively. This is remarkable, considering that COLA is only trained with data consisting of $|\mathcal{V}_H| = \{4, \cdots, 8\}$.

We use Fig. 4 to illustrate an example of COLA's solution quality in comparison with the benchmark. In the example, there are 3 HAs having GUs traffic that needs to be backhauled to the macro-BS. In the Gurobi solution shown in Fig. 4a (where all links are assumed to have perfect LoS), the total number of hops is 12, and 8 UBSs are deployed. In contrast, COLA's solution in Fig. 4b has the same number of hops but deploys 9 UBSs. Although the number of hops is the same for both Gurobi and COLA, we focus on the energy-efficiency of the paths which depends on the number of UBSs deployed. In this example, the total energy consumption by using COLA's suggested paths is 15.90 kJ, while the optimal total energy consumption is 14.50 kJ, which gives an EER of 91.19%. This shows that COLA can suggest feasible paths with close-to-optimal energy cost for a given problem instance. In addition, Fig. 4b shows that with COLA, the constraints of the original MILP problem are satisfied: 1) there are no more than one UBS on a lamppost, 2) traffic from all the source nodes get to the macro-BS, 3) multiple flows can traverse a link (e.g., $\{(19, 24), (24, 0)\}$), and 4) flow conservation is guaranteed in intermediate nodes. This means that the paths suggested by COLA form a feasible solution to the MILP problem.

Since COLA achieves good performance for the LoS scenario in Fig. 4b, we further investigate its performance under challenging non-LoS scenario where a high-rise building entirely blocks the transmission path between two UBSs, creating a "shadow region" behind it where the signal is weak or lost. The macro-BS, which coordinates the deployment of UBSs, also has global network information and can quickly deactivate any non-LoS link (i, j), thus facilitating COLA to avoid such links in future backhauling tasks (i.e. $\hat{W}^d_m[i, j] = \infty$). This is demonstrated in Fig. 4c where link $(19, 24)$ suffers signal blockage and COLA finds an alternate path to the macro-BS. The total energy consumption using COLA for the non-LoS scenario is 16.51 kJ, which gives an EER of 87.82%, and this is achieved without any retraining or fine-tuning.

2) CTR of COLA: The computation time in solving the problem instance increases w.r.t. the number of HAs, both for Gurobi and COLA as shown in Table V. However, with COLA, the computation time is greatly reduced owing to its capability to directly predict the link weights to find the shortest paths, unlike Gurobi which needs to solve the NP-hard MILP problem. In Table V, for $|\mathcal{V}_H| = 10$, COLA achieves

TABLE V: Performance evaluation of COLA for a small-sized network.

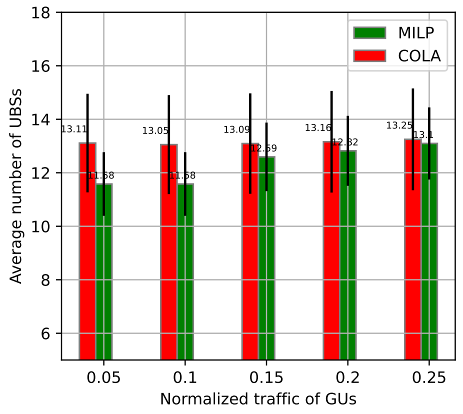
# of HAs	UDA %	EER %	t_{OPT}	t_{COLA}	CTR %
2	88.02	98.51	0.059 ± 0.015	0.003 ± 0.00	95.41
3	90.70	97.35	0.084 ± 0.030	0.004 ± 0.00	95.35
4	92.57	97.45	0.106 ± 0.044	0.005 ± 0.00	95.10
5	86.46	94.96	0.144 ± 0.080	0.006 ± 0.00	95.56
6	88.23	94.56	0.160 ± 0.082	0.008 ± 0.00	95.19
7	87.37	93.64	0.166 ± 0.049	0.009 ± 0.00	94.63
8	86.83	93.24	0.170 ± 0.052	0.010 ± 0.00	94.06
9	83.97	91.36	0.198 ± 0.053	0.012 ± 0.00	94.19
10	84.93	91.11	0.202 ± 0.049	0.013 ± 0.00	93.77

a CTR of 93.77% with $t_{\text{COLA}} = 13$ ms, which means that COLA can swiftly find the near-optimal UBSs to be deployed and construct the mmWave backhauling to the macro-BS, when the number of HAs is as high as 10. Besides that, the computation complexity of COLA is constant as shown by the standard deviation of t_{COLA}, where we have $\pm$ 0.00. The reason for the constant complexity of COLA is because Dijkstra's complexity is $\mathcal{O}(|\mathcal{V}_L|^2)$ (it only increases w.r.t. the number of HAs) which depends on the number of nodes in the network topology, unlike the standard deviation of t_{OPT} which varies stochastically because of the changes in the constraint matrix for different problem instances.

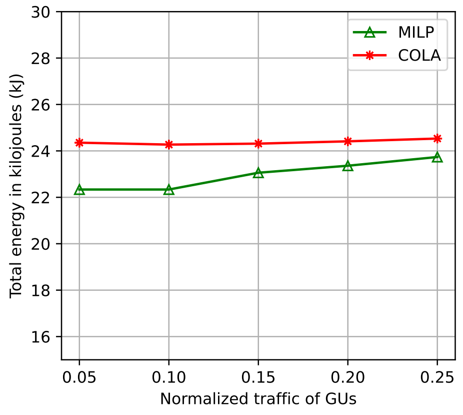
3) Adaptability to Dynamic GU Traffic: In real network environments where traffic demands are spatiotemporal distributions, their routing solutions vary accordingly. For instance, during the daytime, the amount of GUs traffic is usually high (referred to as high-peak period), and at nighttime the amount of traffic is low (referred to as off-peak period). During high-peak periods, the number of UBSs to be deployed increases, which in turn increases the total energy consumption. In Fig. 5, we evaluate COLA's performance when the amount of GUs traffic (i.e., normalized) varies from 0.05 (off-peak) to 0.25 (high-peak). The number of HAs is 8, and the results are the average of 1,000 instances.

For both COLA and the MILP benchmark, the average number of UBSs and energy consumption increase as the amount of GUs' traffic increases, as shown in Fig. 5a and Fig. 5b, respectively. For the off-peak period of 0.05, COLA suggests 13.11 UBSs which is 13% higher than the MILP benchmark. However, this performance gap decreases to 1.18% for the high-peak period of 0.25, thus showing that COLA greatly favors high-peak traffic settings in B5G/6G networks. Regarding energy consumption, COLA achieves a performance close to the MILP benchmark as the amount of GUs traffic increases, as shown in Fig. 5b. When the amount of GUs' traffic is 0.05, COLA achieves 24.35 kJ energy, which is 9.05% higher than the MILP benchmark. On the other hand, when the amount of GUs' traffic increases to 0.25, COLA achieves 24.53 kJ energy, which is only 3.37% higher than the MILP benchmark.

4) COLA vs. Naive ML: In Fig. 6, the performance of COLA is compared with a Naive ML approach which tries to predict whether or not a link in the topology should be active given the application-level input. This also means predicting if a UBS should be deployed to the lampposts to establish those links. This approach trains a multi-label MLP classifier, where each class label is a binary value with 1 denoting an active link and 0 otherwise. The MLP has 2 hidden layers and an output layer with $|\mathcal{E}|$ neurons. The hidden layer neurons use ReLU activation and the output layer neurons use sigmoid activation. Note that this method is naive as it does not find the shortest path solution for the problem. Hence, we can only evaluate its performance based on the UDA metric. From the comparison in Fig. 6, COLA outperforms the Naive ML method in all cases of the number of HAs.



(a) Number of UBSs.



(b) Energy consumption.

Fig. 5: COLA's generalization to dynamic traffic of GUs.

C. Comparison between COLA and DQN

A reinforcement learning (RL) approach that uses a deep Q-network (DQN) with experience replay [33] is compared to COLA. The state space, action space, and reward function are defined as follows:

TABLE VI: Comparing COLA and DQN for a small-sized network.

# of HAs	2	3	4	5	6	7	8	9	10
UDA_{DQN}	32.66	30.17	27.70	28.71	28.46	22.96	24.71	24.65	26.35
UDA_{COLA}	80.67	88.45	87.81	83.00	82.24	85.21	83.02	81.41	80.81
EER_{DQN}	61.23	61.91	65.80	66.90	64.93	64.44	65.89	69.11	69.05
EER_{COLA}	93.08	94.99	92.62	93.00	89.76	91.53	90.17	89.80	88.37
CTR_{DQN}	70.41	69.57	70.06	69.97	68.54	64.67	60.96	61.57	57.69
CTR_{COLA}	94.22	95.23	95.10	95.49	95.13	94.50	94.00	94.29	93.72

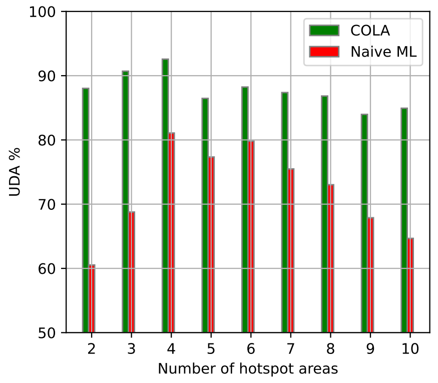


Fig. 6: UDA comparison between COLA and Naive ML for the small-sized network.

1) State Space: This reflects the environment observed by the agent (i.e., the current UAV). The state space is defined as $S_t = (A_m^d, \mathbf{1}_u)$, where A_m^d is the application-level information for a single-commodity flow (as in eq. 18) and $\mathbf{1}_u$ is the one-hot vector that represents the current UAV u.

2) Action Space: The action space consists of the UAV's neighbors, i.e., $A_t = (u' \mid u' \in N(u))$, where $N(u)$ are the neighbors of UAV u. At each state S_t, the agent (i.e., UAV) can select an action $u' \in A_t$ corresponding to the next hop.

3) Reward Function: Since the goal is to reach the destination macro-BS with minimum energy consumption, the reward R_t is defined as:

$$R_t = \begin{cases} e_{opt}^{total} - \sum_{u' \in P} e_{u'}, & \text{if macro-BS is reached} \\ -\rho e_{u'}, & \text{if } u' \in P \text{ or } u' \neq \text{macro-BS when } t = T_{\max} \\ -e_{u'}, & \text{otherwise} \end{cases}$$

where e_{opt}^{total} is the optimal energy obtained from the MILP solver which serves as a "reward" signal in finding energy-efficient paths and $e_{u'}$ is the energy consumption by UAV u' when selected from the action space. There is a ρ term ($\rho = 5$ in our setting) in the reward function that is used to penalize the algorithm if the selected action u' results in a loop or if the algorithm does not find the path within the maximum steps $T_{\max}$ required. Agents select actions based on the ϵ-greedy policy. The parameters of the DQN are those used in [11] and [33].

The training data used for the performance comparison between COLA and DQN consists of 10,000 instances for each $|\mathcal{V}_H| \in \{4, \cdots, 8\}$ making a total of 50,000 instances. We consider the small-sized network for the comparison. Similar to COLA, we apply the data decomposition technique to obtain single-commodity flow samples to facilitate the DQN. Table VI shows the results of using 1,000 instances for each $|\mathcal{V}_H| \in \{2, \cdots, 10\}$. This means that both COLA and DQN are evaluated on smaller and larger numbers of HAs that are different from those used in training.

The result of the DQN in Table VI shows the limitation of RL in achieving a close-to-optimal solution. First, the UDA score is low since there is no reward for selecting the exact nodes as the optimal solution (this is because the optimal solution itself is not always unique), but the COLA loss function helps mitigate this issue. Second, the EER obtained by using the DQN-based routing method is low compared to COLA, because DQN requires a large number of routing scenarios which is time-consuming to train. This is not the case with COLA: by using the same limited number of training instances, its performance on the EER metric is significantly higher than DQN. Third, the computational cost incurred in running the DQN algorithm during inference is higher than that of COLA, expressed by the CTR metric (the higher the better). In contrast to COLA which directly outputs the suggested path from source to destination in one round, the DQN-based routing method requires several rounds (corresponding to the number of hops) to compute the path.

In summary, the UAV-assisted backhauling is a challenging problem for RL methods (e.g., DQN) to solve. To achieve a performance close to COLA, the RL agents (i.e., UAVs) need to collaborate with each other to know how their local decisions affect the global energy consumption. This will require huge communication and computation overhead w.r.t. the training time for the algorithm to converge.

D. Performance in a Large-sized Network

The large-sized network is an 8×8 grid topology which consists of 63 lampposts uniformly positioned within the 400 m by 400 m layout and a single macro-BS located on the top-right corner in the layout as illustrated in Fig. 3b. The maximum number of UBSs is 63. The training data consists of 83,000 instances for only $|\mathcal{V}_H| = 10$ and the HAs are random with different normalized traffic for GUs. The test data consists of 1,000 different instances for each $|\mathcal{V}_H| \in \{3, 4, 5, 10, 15, \cdots, 20\}$. This means that COLA was trained only with instances having 10 flow demands, but we evaluated its performance on smaller flow demands as well as very large flow demands (up to 20 HAs).

As shown in Table VII, COLA achieves a UDA of up to 75.32% and this is for the case where there are 19 HAs. The EER is up to 89.15% for $|\mathcal{V}_H| = 3$ and the minimum is 79.24% for $|\mathcal{V}_H| = 18$. We observe that there is not much difference in the UDA and EER of smaller and larger $|\mathcal{V}_H|$, and this

TABLE VII: Performance evaluation of COLA for a large-sized network.

# of HAs	UDA %	EER %	t_{OPT}	t_{COLA}	CTR %
3	75.31	89.15	0.896 ± 0.792	0.016 ± 0.00	98.23
4	73.24	88.79	1.262 ± 0.818	0.021 ± 0.00	98.33
5	72.50	85.66	1.784 ± 1.480	0.026 ± 0.00	98.54
10	72.79	82.12	5.761 ± 4.974	0.056 ± 0.00	99.02
15	74.79	81.03	7.388 ± 7.176	0.079 ± 0.00	98.94
16	72.38	79.68	7.356 ± 6.935	0.083 ± 0.00	98.87
17	73.81	80.03	6.985 ± 5.846	0.089 ± 0.00	98.72
18	72.14	79.24	6.927 ± 4.559	0.096 ± 0.00	98.61
19	75.32	80.16	6.029 ± 3.677	0.099 ± 0.00	98.36
20	75.24	80.75	6.168 ± 4.071	0.106 ± 0.00	98.27

is because the COLA has learned to generalize to unseen instances, as evidenced by the UDA and EER performance in the extremely complex scenario of $|\mathcal{V}_H| = 20$ (the UDA is higher and the EER is close to that of $|\mathcal{V}_H| = 10$ which COLA is trained with). Comparing the computation time of COLA and Gurobi, we show in Table VII that for $|\mathcal{V}_H| = 15$, while t_{OPT} is 7.388 seconds, t_{COLA} is only 79 ms, yielding a CTR of 98.94%. Similar to its performance in the small-sized network, the time complexity of COLA is constant for the problem instances, as the standard deviation of t_{COLA} is ± 0.00 for all $|\mathcal{V}_H|$. For the small-sized network and large-sized network, the computation time of COLA is in milliseconds (ms), with the highest being 106 ms for $|\mathcal{V}_H| = 20$ in the large-sized network. Hence, COLA offers a promising benefit of achieving low-latency backhauling in B5G/6G wireless networks.

It is worth mentioning that the performance comparison of COLA with Naive ML for the large-sized network is similar to that of the small-sized network shown in Fig. 6. However, due to the page limitation of this paper, we do not further show this comparison.

E. Ablation Study of COLA

We conducted some ablative experiments to provide insights on the impact of the different components of COLA on the UDA metric. Here, we also use the PMA metric for comparison. It measures how perfect the suggested path is to the optimal path and it is the higher the better. For the ablation study of COLA, the performance is only evaluated for the small-sized network and the training is set to 150 epochs. The components we examine are:

1) Number of Neural Network Layers: We evaluate the effect of the number of layers on COLA's performance. As shown in Table VIII, when the number of layers is reduced from four to one, the UDA and PMA greatly improve from 70.56% and 39.96% to 83.22% and 62.18%, respectively, comparing row 1 and row 7. This is because with multiple layers in MLP, the gradients become progressively smaller as they are being passed from the last layer to the first layer in the MLP.

2) Activation Function: The choice of the activation function plays a critical role in training COLA. As shown in Table VIII, for the case of three layers in row 4, we observe that using ReLU activation causes the UDA and PMA to significantly drop by 37% and 59%, respectively, relative to the performance when Linear activation is used in row 3. While Linear activation can handle derivatives of negative inputs, ReLU takes the derivative of negative inputs as zero. Hence, during backpropagation, most gradients are zero for ReLU and the neural network parameters no longer update.

3) Batch Selection: We compare two methods for selecting batches from the dataset. The *random* selection method randomly orders the batches for each training epoch, whereas the *fixed* selection method uses the same order of the batches for each training epoch. In Table VIII, the fixed batch selection method in row 11 outperforms the random batch selection method in row 8. The rationale behind this is that, with the fixed method, the neural network parameters are updated progressively over the training epochs, unlike the random method which causes inconsistent parameter updates and slower convergence. Assuming at the first epoch the batch order is $\{1, 2, 3, 4\}$ and at the second epoch, the batch order changes to $\{4, 1, 3, 2\}$, there will be oscillations in the gradient flow due to the randomness in the batch selection.

4) Batch Size: The size of the data batch is an important hyperparameter for COLA. From the experiments, we observe that using a batch size of 302 gives the best UDA and PMA. Increasing the batch size to 817 or reducing it to 151 reduces COLA's performance as shown in rows 7 to 9 of Table VIII for comparison.

5) Loss Function: Defining an appropriate loss function can greatly influence COLA's performance as shown in Table VIII (rows 6 and 7), where using the MSE loss achieves a better performance than the hamming loss (HL). With the MSE loss, we achieve a UDA and PMA of 83.22% and 62.18%, respectively, whereas for HL, the UDA and PMA are 73.70% and 44.63%, respectively. The HL does not provide useful gradients during backpropagation despite using its relaxed definition as follows:

$$\text{HL} = \frac{1}{|\mathcal{V}_L|^2} \sum_{i=1}^{|\mathcal{V}_L|} \sum_{j=1}^{|\mathcal{V}_L|} P_m^d[i,j] \cdot (1 - \hat{P}_m^d[i,j]) + \hat{P}_m^d[i,j] \cdot (1 - P_m^d[i,j]) \quad (30)$$

This means that the MSE is a better loss function for the implementation of COLA.

6) Perturbation with λ: We compare the performance of COLA when a fixed λ value is used to perturb the link weights and when an adaptive λ value is used. In row 11 and row 12 of Table VIII, we show that setting a fixed λ value achieves a performance of 87.20% and 69.76% for the UDA and PMA, respectively, whereas with an adaptive λ (using ϵ_λ as control parameter), the UDA and PMA increase to 89.19% and

TABLE VIII: Ablation study of COLA.

Row ID	# of Layers	Activation	Batch selection	Batch size	Loss function	Perturbation (λ)	UDA %	PMA %
1	Four	Linear	Random	151	MSE	Fixed	70.56	39.96
2	Four	Linear	Random	151	MSE	Fixed	74.23	44.87
3	Three	Linear	Random	151	MSE	Fixed	72.86	44.87
4	Three	ReLU	Random	151	MSE	Fixed	46.01	18.36
5	Two	Linear	Random	151	MSE	Fixed	79.76	55.98
6	One	Linear	Random	151	HL	Fixed	73.7	44.63
7	One	Linear	Random	151	MSE	Fixed	83.22	62.18
8	One	Linear	Random	302	MSE	Fixed	85.24	65.98
9	One	Linear	Random	817	MSE	Fixed	81.50	57.81
10	One	Linear	Random	302	MSE	Adaptive	84.98	66.70
11	One	Linear	Fixed	302	MSE	Fixed	87.20	69.76
12	One	Linear	Fixed	302	MSE	Adaptive	**89.19**	**73.22**

73.13%, respectively. With extensive experiments, we found that setting $\epsilon_\lambda \approx 0.2\lambda$ achieves better training stability (see Table IV for actual value).

To further compare performance, we used a subset of the training data and run COLA for over 1000 epochs to observe the convergence based on fixed λ and adaptive λ. According to Fig. 7, setting a fixed λ halts the training after about 190 epochs as there is no improvement in the loss, whereas with adaptive λ, the training loss continues to decrease, leading to a better convergence, but at a cost of more training epochs.

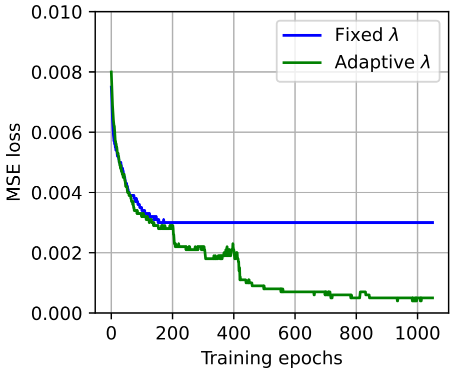


Fig. 7: Convergence w.r.t. perturbation hyperparameter.

The results from the ablation study reveal that using an SLP with Linear activation function and MSE loss, in combination with a fixed batch selection (and batch size of 302), while using an adaptive λ perturbation hyperparameter, gives the best performance for COLA, as shown in row 12 of Table VIII.

F. Applicability of COLA and the Practical Challenges in 6G Non-Terrestrial Networks

In 6G non-terrestrial networks (NTNs), UAVs move rapidly, causing intermittent topology changes. This affects the reliability of the network due to unstable links [34]. Furthermore, the power constraints of the UAVs make an energy-efficient communication protocol a necessity. The implementation of COLA for 6G NTNs has potential to address the problem of dynamic topology changes, but with a limitation on additional factors such as interference, very high mobility of GUs, and unpredictable weather conditions. COLA's remarkable performance in maintaining energy-efficient communication among UBSs in both small and large networks offers promising insights in mitigating the UAV power constraint problem in 6G NTNs.

The strict latency requirement for time-sensitive applications remains a factor to be addressed. Although COLA achieves a latency of 106 ms for backhauling in the large-sized network, a latency of ∼30 to 50 ms is the expectation in 6G NTNs. This remains a research direction to be explored, but we recommend using lightweight hardware accelerators for COLA to achieve improved latency. On the other hand, COLA's ability to solve very complex scenarios in 6G NTNs (e.g., dynamic channel conditions, spectrum sharing, and beam coordination) still needs to be studied. We envision that COLA's modular structure can be extended to account for such dynamics, for instance, by integrating real-time link state estimations into the neural network input.

VI. Related Work

A. State of the Art of UAV-Assisted Wireless Backhauling

The conventional FBSs wireless backhauling approach suffers drawbacks in terms of the installation and operating costs of the FBSs [8], especially when considering the dynamic wireless network environment where the traffic load of small cells changes stochastically in space and time. Also, when natural disasters happen, GUs get disconnected from the terrestrial network as a result of damaged FBSs. To effectively restore network connectivity, UAVs need to be quickly deployed to the affected disaster areas. Recent studies have examined UAVs in emergency communication networks. The work in [35] focused on the joint optimization of the macro-BS power allocation, UAV service zone selection, and user scheduling to improve the sum spectrum efficiency. In a bid to achieve throughput fairness among GUs, the work in [36] investigated the 3-D UAV trajectory design and band allocation problem, with an objective to maximize total throughput and fairness with the minimum energy possible. Several other works [37]–[40] have also addressed similar problems related to UAV location, flight trajectory, and resource allocation from an optimization perspective. With the flying ability of UAVs, they can be integrated into the wireless network infrastructure, to assist the backhauling of GUs' data to the macro-BS.

Some benefits of leveraging UAVs for wireless backhauling in B5G/6G networks have been investigated in the literature [9], [18]. The work in [9] proposed the concept of UAVs as robotic aerial small cells (RASCs) in 6G wireless networks, where the RASCs are equipped with grasping end effectors to grasp tall buildings and lampposts to enhance connectivity and serve as relay backhauling nodes to the macro-BS. The study shows that the RASCs can assist in reducing the energy consumption of wireless backhauling compared to the use of FBSs. The work in [18] extends [9] by focusing on the RASCs serving both as an access point for the GUs and as relay nodes for wireless backhauling. Both [9] and [18] formulated an MILP problem to minimize the number of RASCs, the number of hops, and optimize the total energy consumption, although [18] includes the maximization of the traffic demand on each route. While the work in [9] solved the MILP problem with an optimization solver in MATLAB, [18] proposed a greedy algorithm to find the near-optimal solutions.

Prior to [9], [18], the works in [14], [15], [41] had considered UAVs for wireless communications, where the UBSs are used to transmit data from GUs to the macro-BS by optimizing the flight time and flight path to minimize the power consumption and maximize the total harvested energy by the UBSs. The system model is such that a single UBS is deployed to collect data from the GUs in different locations and fly back (while harvesting solar energy [14] along the flight path) to offload the data to the macro-BS [15], [41]. However, the works in [9], [18] opined that multiple RASCs, rather than a single UBS, can be flexibly deployed to perform energy-efficient wireless backhauling in dense urban areas, while providing wireless access to GUs.

B. Machine Learning for Network Flow Optimization

Known for its rich capability in learning complex data structure and making predictions, ML has been a powerful tool in wireless networking research in addressing challenges in the areas of resource scheduling [19], [25], [42], routing [43]–[45], mobility [44], and localization [44], [46]. Several works [19], [25], [42] have proposed ML frameworks to tackle the computational complexity issues of traditional algorithms in solving NP-hard problems like network flow optimization. The work in [19] proposed a self-supervised ML approach for accelerating the optimization of wireless network capacity for integrated access and backhauling (IAB) in B5G/6G networks, where the scheduling and forwarding are coupled in the network flow problem formulation. Similarly, ML has been exploited to achieve significant performance gain in wireless network optimization, such as leveraging a deep learning (DL) approach to reduce the problem scale of a flow constrained optimization problem by predicting the set of unused links to be removed from the topology before running the linear programming for a problem instance [42]. RL techniques have also been proposed as a viable approach for wireless network optimization. The work in [47] proposed a deep RL (DRL) algorithm for the independent set scheduling in wireless networks, while [35], [37], [40] proposed a DRL approach for UAV zone selection, power allocation, and sum spectrum efficiency.

Prior to COLA, convolutional neural networks (CNN), recurrent neural networks, and DRL have been proposed to address routing problems in wireless networks [33], [44], [48], [49]. The work in [48] proposed a deep CNN-based method for intelligent routing decisions in a wireless backbone network consisting of several routers, but the proposed method is not practically scalable to large-sized networks as it requires training the deep learning model in an online fashion. The work in [33] proposed a DRL-based routing strategy in a flying adhoc network (FANET). Specifically, a DQN is trained to select the next hop node in the FANET according to the Q-values predicted by the model [33]. The work in [50] proposed an RL approach to improve the achievable data rate in an IAB network, where the UAVs are integrated to enhance wireless connectivity and traffic forwarding in mission-critical and high-risk situations.

DRL methods have largely been proposed for UAV-assisted wireless backhauling. The works in [11], [43], [50]–[52] consider the role UAVs play in B5G/6G networks in the context of enhancing backhauling when the users' locations change and the traffic demand varies. Specifically, [51] designed a DRL algorithm to solve the network flow optimization problem, where the objective is to jointly optimize the 3-D locations of the UBSs and the flight path to accommodate the dynamically changing user distribution, while maximizing the backhaul link rate. This is similar to the work in [11] which proposed a dueling double deep Q-network (D3QN) to optimize UAV placement in dynamically changing traffic conditions in a 5G IAB network, while striking a balance between fronthaul and backhaul connectivity. The work in [43] also proposed Q-learning geographic-based routing algorithm to improve the network performance of UAVs in high-mobility scenarios. The objective is to achieve high-packet delivery ratio by selecting the best routes for the UAVs, while reducing the network overhead.

It is worthy to note that RL methods are particularly beneficial when the decision making is sequential, which is what most of the RL-related works on wireless backhauling have proposed. However, for large-scale dynamic network management where the route planning is important before traffic forwarding, DRL cannot provide scalable solutions due to the state space and action space explosion as the number of nodes, links, and flows increases. Another issue is the reward design - an RL algorithm needs to find the routing paths for all the source-destination pairs of the multi-commodity flows before providing rewards, which can hinder effective learning in the sense of sparse rewards. Likewise, the solution time of RL algorithms in finding the sequential hops for a network flow to the destination can be high if the agent is interacting in an unknown environment that is very different from the environment it observed during training.

With the prevailing issues of conventional ML approaches, in this paper, we propose the combinatorial optimization learning paradigm for UAV-assisted wireless backhauling, which has not been explored before in the literature, to the best of our knowledge. The closest to our work is [28], [53], where the fusion of DL with combinatorial optimization is proposed, by leveraging a DL model to predict the values of one of the decision variables in a linear optimization problem, while the optimization solver computes the remaining decision variables.

This only reduces the size of the constraint matrix and the solution-search space. Also, the studies considered problems that are not related to wireless networking, which makes them greatly different from the complex wireless backhauling presented in this paper.

VII. Conclusion

This paper has considered mmWave backhauling in UAV-assisted wireless networks by proposing COLA to solve the multi-commodity flow optimization problem which is well-known to be NP-hard. At any given time, when traffic from multiple GU locations needs to be backhauled to the macro-BS, COLA can directly suggest the energy-efficient paths with greatly reduced computation cost. The adoption of COLA for the MILP problem in this paper is important, considering the next generation of wireless networks, which is envisioned to be artificial intelligence-driven at multiple levels. The COLA in this paper provides clear insights on the methodology of leveraging ML as a disruptive tool to solve complex combinatorial optimization problems in wireless networking, to meet the requirements of reliability and efficiency.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under Grant CNS-2008092.

References

[1] S. Andreev, V. Petrov, M. Dohler, and H. Yanikomeroglu, “Future of ultra-dense networks beyond 5G: Harnessing heterogeneous moving cells,” *IEEE Communications Magazine*, vol. 57, no. 6, pp. 86–92, 2019.

[2] M. Latva-Aho, K. Leppänen *et al.*, “Key drivers and research challenges for 6G ubiquitous wireless intelligence,” 2019.

[3] A. Alhammadi, I. Shayea, A. A. El-Saleh, M. H. Azmi, Z. H. Ismail, L. Kouhalvandi, and S. A. Saad, “Artificial intelligence in 6G wireless networks: Opportunities, applications, and challenges,” *International Journal of Intelligent Systems*, vol. 2024, no. 1, p. 8845070, 2024.

[4] P.-H. Huang and K. Psounis, “Optimal backhauling for dense small-cell deployments using mmWave links,” *Computer Communications*, vol. 138, pp. 32–44, 2019.

[5] J. McMenamy, A. Narbudowicz, K. Niotaki, and I. Macaluso, “Hop-constrained mmWave backhaul: Maximising the network flow,” *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 596–600, 2019.

[6] Y. Yan, Q. Hu, and D. M. Blough, “Feasibility of multipath construction in mmWave backhaul,” in *Proc. of IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2021, pp. 81–90.

[7] A. H. Jafari, D. López-Pérez, H. Song, H. Claussen, L. Ho, and J. Zhang, “Small cell backhaul: challenges and prospective solutions,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, pp. 1–18, 2015.

[8] C. Bouras, V. Kokkinos, and A. Papazois, “Financing and pricing small cells in next-generation mobile networks,” in *Wired/Wireless Internet Communications: 12th International Conference*, 2014, pp. 41–54.

[9] J. Lee and V. Friderikos, “Robotic aerial 6G small cells with grasping end effectors for mmWave relay backhauling,” in *Proc. of IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2022, pp. 1–6.

[10] I. Bor-Yaliniz and H. Yanikomeroglu, “The new frontier in RAN heterogeneity: Multi-tier drone-cells,” *IEEE Communications Magazine*, vol. 54, no. 11, pp. 48–55, 2016.

[11] Y. Wang and J. Farooq, “Deep reinforcement learning based placement for integrated access backhauling in UAV-assisted wireless networks,” *IEEE Internet of Things Journal*, 2023.

[12] Y. Zhang, H. Shan, M. Song, H. H. Yang, X. Shen, Q. Zhang, and X. He, “Packet-level throughput analysis and energy efficiency optimization for UAV-assisted iab heterogeneous cellular networks,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9511–9526, 2023.

[13] B. Galkin, J. Kibilda, and L. A. DaSilva, “UAVs as mobile infrastructure: Addressing battery lifetime,” *IEEE Communications Magazine*, vol. 57, no. 6, pp. 132–137, 2019.

[14] H. D. Tuan, A. A. Nasir, A. V. Savkin, H. V. Poor, and E. Dutkiewicz, “MPC-based UAV navigation for simultaneous solar-energy harvesting and two-way communications,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3459–3474, 2021.

[15] D. Wu, X. Sun, and N. Ansari, “An fso-based drone charging system for emergency communications,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 155–16 162, 2020.

[16] M. T. Dabiri and M. Hasna, “3D uplink channel modeling of UAV-based mmWave fronthaul links for future small cell networks,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 1400–1413, 2022.

[17] K. Messaoudi, O. S. Oubbati, A. Rachedi, A. Lakas, T. Bendouma, and N. Chaib, “A survey of UAV-based data collection: Challenges, solutions and future perspectives,” *Journal of network and computer applications*, vol. 216, p. 103670, 2023.

[18] W. Shang, Y. Liao, V. Friderikos, and H. Yanikomeroglu, “Joint robotic aerial base station deployment and wireless backhauling in 6G multi-hop networks,” *arXiv preprint arXiv:2405.07714*, 2024.

[19] O. T. Ajayi, S. Zhang, and Y. Cheng, “Machine learning assisted capacity optimization for B5G/6G integrated access and backhaul networks,” in *Proc. of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–6.

[20] S. Zhang, B. Yin, W. Zhang, and Y. Cheng, “Topology aware deep learning for wireless network optimization,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 11, pp. 9791–9805, 2022.

[21] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, “Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1251–1275, 2020.

[22] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, “A survey on machine learning techniques for routing optimization in SDN,” *IEEE Access*, vol. 9, pp. 104 582–104 611, 2021.

[23] S. Zhang, B. Yin, S. Wang, and Y. Cheng, “Robust deep learning for wireless network optimization,” in *Proc. of IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[24] W. Cui, K. Shen, and W. Yu, “Spatial deep learning for wireless scheduling,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.

[25] S. Zhang, O. T. Ajayi, and Y. Cheng, “A self-supervised learning approach for accelerating wireless network optimization,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 8074–8087, 2023.

[26] V. Friderikos, “Airborne urban microcells with grasping end effectors: A game changer for 6G networks?” in *Proc. of IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021, pp. 336–341.

[27] M. Höyhtyä, O. Apilo, and M. Lasanen, “Review of latest advances in 3GPP standardization: D2D communication in 5G systems and its energy consumption models,” *Future Internet*, vol. 10, no. 1, p. 3, 2018.

[28] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek, “Differentiation of blackbox combinatorial solvers,” in *International Conference on Learning Representations*, 2020.

[29] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder, “End-to-end constrained optimization learning: A survey,” *arXiv preprint arXiv:2103.16378*, 2021.

[30] E. W. Dijkstra, “A note on two problems in connexion with graphs,” in *Edsger Wybe Dijkstra: his life, work, and legacy*, 2022, pp. 287–290.

[31] C. Liu, M. Ding, C. Ma, Q. Li, Z. Lin, and Y.-C. Liang, “Performance analysis for practical unmanned aerial vehicle networks with LoS/NLoS transmissions,” in *Proc. of IEEE international conference on communications workshops (ICC workshops)*, 2018, pp. 1–6.

[32] Y. Zeng, J. Xu, and R. Zhang, “Energy minimization for wireless communication with rotary-wing UAV,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[33] D. Lin, T. Peng, P. Zuo, and W. Wang, “Deep-reinforcement-learning-based intelligent routing strategy for FANETs,” *Symmetry*, vol. 14, no. 9, p. 1787, 2022.

[34] F. Wang, S. Zhang, H. Yang, and T. Q. Quek, “Non-terrestrial networking for 6G: Evolution, opportunities, and future directions,” *Engineering*, 2025.

[35] C. Wang, D. Deng, L. Xu, and W. Wang, “Resource scheduling based on deep reinforcement learning in UAV assisted emergency communication networks,” *IEEE Transactions on Communications*, vol. 70, no. 6, pp. 3834–3848, 2022.

[36] R. Ding, F. Gao, and X. S. Shen, "3D UAV trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7796–7809, 2020.

[37] W. Huang, Z. Yang, C. Pan, L. Pei, M. Chen, M. Shikh-Bahaei, M. Elkashlan, and A. Nallanathan, "Joint power, altitude, location and bandwidth optimization for UAV with underlaid D2D communications," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 524–527, 2018.

[38] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, "Optimal 3d-trajectory design and resource allocation for solar-powered UAV communication systems," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4281–4298, 2019.

[39] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[40] J.-M. Kang and C.-J. Chun, "Joint trajectory design, Tx power allocation, and Rx power splitting for UAV-enabled multicasting SWIPT systems," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3740–3743, 2020.

[41] M. Kishk, A. Bader, and M.-S. Alouini, "Aerial base station deployment in 6G cellular networks using tethered drones: The mobility and endurance tradeoff," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 103–111, 2020.

[42] L. Liu, B. Yin, S. Zhang, X. Cao, and Y. Cheng, "Deep learning meets wireless network optimization: Identify critical links," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 167–180, 2018.

[43] W.-S. Jung, J. Yim, and Y.-B. Ko, "QGeo: Q-learning-based geographic ad hoc routing protocol for unmanned robotic networks," *IEEE Communications Letters*, vol. 21, no. 10, pp. 2258–2261, 2017.

[44] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[45] C. V. Murudkar and R. D. Gitlin, "Optimal-capacity, shortest path routing in self-organizing 5G networks using machine learning," in *Proc. of IEEE 20th Wireless and Microwave Technology Conference (WAMICON)*, 2019, pp. 1–5.

[46] D. Li, B. Zhang, and C. Li, "A feature-scaling-based k-nearest neighbor algorithm for indoor positioning systems," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 590–597, 2015.

[47] S. Zhang, B. Yin, and Y. Cheng, "Deep reinforcement learning for scheduling in multi-hop wireless networks," in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2021, pp. 1–9.

[48] F. Tang, B. Mao, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 154–160, 2017.

[49] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 146–153, 2016.

[50] N. Tafintsev, D. Moltchanov, M. Simsek, S.-P. Yeh, S. Andreev, Y. Koucheryavy, and M. Valkama, "Reinforcement learning for improved UAV-based integrated access and backhaul operation," in *Proc. of IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–7.

[51] H. Zhang, Z. Qi, J. Li, A. Aronsson, J. Bosch, and H. H. Olsson, "5G network on wings: A deep reinforcement learning approach to the UAV-based integrated access and backhaul," *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[52] M. Jaber, M. Imran, R. Tafazolli, and A. Tukmanov, "An adaptive backhaul-aware cell range extension approach," in *Proc. of IEEE international conference on communication workshop (ICCW)*, 2015, pp. 74–79.

[53] J. Mandi, P. J. Stuckey, T. Guns *et al.*, "Smart predict-and-optimize for hard combinatorial optimization problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 1603–1610.

Oluwaseun T. Ajayi (Student Member, IEEE) received the B.S. degree in telecommunication science from the University of Ilorin, Nigeria, in 2018 and the Master's degree in electrical and computer engineering from the Illinois Institute of Technology in 2024. He is currently pursuing his Ph.D. in electrical engineering at the Illinois Institute of Technology. His research interests include machine learning, wireless network optimization, information freshness, cloud/edge computing, and AI for science.

Suyang Wang (Member, IEEE) received the B.E. degree in telecommunications engineering from Xidian University in 2015, and the M.S. and Ph.D. degrees in electrical engineering from the Illinois Institute of Technology in 2017 and 2024, respectively. He is currently an Assistant Professor in the School of Computer Science and Engineering at California State University, San Bernardino. His research interests include information freshness, wireless network optimization, generative AI, machine learning, network security, and blockchain.

Yu Cheng (Fellow, IEEE) received B.E. and M.E. degrees in electronic engineering from Tsinghua University in 1995 and 1998, respectively, and a Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2003. He is now a full professor in the Department of Electrical and Computer Engineering, Illinois Institute of Technology. His research interests include wireless network performance analysis, information freshness, machine learning, and network security. He received a Best Paper Award at IEEE/CIC ICCC 2024, QShine 2007, IEEE ICC 2011, and a Runner-Up Best Paper Award at ACM MobiHoc 2014. He received the National Science Foundation (NSF) CAREER Award in 2011 and IIT Sigma Xi Research Award in the junior faculty division in 2013. He has served as several Symposium Co-Chairs for IEEE ICC and IEEE GLOBECOM, and Technical Program Committee (TPC) Co-Chair for IEEE MASS 2025, IEEE/CIC ICCC 2015, ICNC 2015, and WASA 2011. He was a founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He was an IEEE ComSoc distinguished lecturer in 2016-2017. He is an Associate Editor for IEEE Transactions on Vehicular Technology, IEEE Internet of Things Journal, and IEEE Wireless Communications. He is an IEEE Fellow.