IEEE Sensors Council

A Comprehensive Planning Framework for Connected Elevated LiDAR Sensors

Nawfal Guefrachi, *Member, IEEE*, Michael C. Lucic, Mohammad Yassen, Hakim Ghazzai, *Senior Member, IEEE*, and Ahmad Alsharoa, *Senior Member, IEEE*

***Abstract*—The combination of mobile edge computing (MEC) and sensing technologies, such as light detection and ranging (LiDAR), offers a viable path toward enhancing autonomous vehicle navigation and traffic monitoring in the context of intelligent transportation systems. In order to meet these needs, this article offers a methodology that investigates the use of elevated LiDAR (ELiD) and its integration with MEC. Our work focuses on two main challenges: optimizing the placement of ELiDs to ensure extensive urban coverage and minimizing network latency by efficiently routing data to MEC servers. By proposing a heuristic for real-time task allocation, we aim to enhance safety and operational efficiency in smart cities. Our findings show a modest optimality gap where the heuristic achieves a balance between computational efficiency and minimized cloud dependency, albeit at the cost of a marginally increased latency, highlighting the nuanced tradeoffs in edge-to-cloud task distribution for efficient LiDAR data processing in smart cities.**

***Index Terms*—Elevated light detection and ranging (LiDAR), infrastructure planning, intelligent transportation systems (ITS), mobile edge computing (MEC), optimization.**

List of the Mathematical Notations and Their Descriptions

Notation	*Definition*
λ, Λ	ELiD, set of ELiDs.
r, $\mathcal{R}$	Roadway, set of roadways.
θ	Horizontal FoV.
ϕ	Vertical FoV.
f_{scan}	Scan refresh rate.
$x^{\lambda r}$	Location of ELiD.
$x_1^{\lambda r}$	ELiD's latitude.
$x_2^{\lambda r}$	ELiD's longitude.
$x_3^{\lambda r}$	ELiD's altitude.
$\rho_1^{\lambda r}$	ELiD's rotation relative to the axis orthogonal to the ground.
$\rho_2^{\lambda r}$	ELiD's rotation relative to the perpendicular roadway plane's normal vector.
D^r	Road's length.
y^r_{min}	ELiD's fixed distance from the roadway.
y^r_{max}	Distance from the side of the potential ELiD location to the far curb on the other side of the street.
τ_t	Road's angle relative to east.
g, $\mathcal{G}$	Section, set of sections.
I_{rg}	Importance metric.
$A^{\lambda r}_{\text{cov}}$	Effective coverage area.
δ	Scan depth parameter.
$\Gamma^{\lambda r}$	Max data produced per scan.
$\Delta^{\lambda r}$	Data produced per scan.
z_{cov}	Predetermined height of the coverage prism.
$E^{\lambda r}$	Joint RSU-ELiD apparatus.
$E^{\lambda r}$	RSU energy consumption.
$E^{\lambda r}_{\text{ELiD}}$	Amount of energy required to power the ELiD scan laser.
P_{trans}	Transmission power.
R_{trans}	Transmission rate.
c, C	Computer network node, set of computer network nodes.
$l_{ij}, \mathcal{L}$	Fiber-optic link, set of fiber-optic links.
$(i, j), \mathcal{V}$	Nodes, set of nodes.
T_{ij}	Link data throughput.
$\mathcal{T}$	Set of directions.
γ	Fraction of the initial map.
$\Delta^{\lambda r}_c$	Size if processed map.

Received 7 June 2024; revised 2 November 2024 and 15 November 2024; accepted 21 November 2024. Date of publication 25 November 2024; date of current version 15 January 2025. Recommended by Lead Guest Editor Dr. Hossein Fotouhi and Guest Editor Dr. Reza Malekian. *(Corresponding author: Ahmad Alsharoa.)*

Nawfal Guefrachi and Ahmad Alsharoa are with the Missouri University of Science and Technology (MST), Rolla, MO 65409 USA (e-mail: aalsharoa@mst.edu).

Michael C. Lucic is with the School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ 07030 USA.

Mohammad Yassen is with the Hashemite University, Zarqa 13133, Jordan.

Hakim Ghazzai is with the CEMSE Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia.

Digital Object Identifier 10.1109/JSAS.2024.3506478

I. Introduction

RECENT developments in embedded systems and sensor technologies have opened up a wide range of applications in the field of intelligent transportation systems (ITS) [1]. Light detection and ranging (LiDAR) is one crucial sensing technique that could greatly improve ITS sensing capabilities [2]. LiDAR uses laser technology to measure the distance between the sensor aperture and the target object of the laser beam. Multiple laser projections over a field of view (FoV) region can be used to build 3-D virtual maps of the vicinity around the LiDAR unit [3].

LiDAR technology has become an essential sensing tool in many ITS-related domains, including autonomous vehicles (AVs) [4] and urban mapping/surveying [5], [6]. It is also remarkable for its crucial role in helping AVs with the simultaneous location and mapping process for autonomous navigation in a variety of environments [7], [8], [9]. The ability of LiDAR systems to deliver detailed data enables the creation of precise 3-D maps of an AV's environment, facilitating more accurate navigation. Despite its significance, LiDAR technology also presents challenges, including the need for large local data storage and processing resources to support AV operations. In specific scenarios, the data generation by AVs can reach substantial levels. For instance, during continuous operation over an 8-h period each day, it is possible for these vehicles to produce a volume of data nearing 40 TB. This volume breaks down to an approximate rate of 84 MB per minute. Therefore, to handle this substantial volume of data, AVs are typically equipped with onboard graphics processing units (GPUs). This approach is adopted because LiDAR data, resembling image data, can utilize established, efficient vectorized image processing methods that take advantage of GPUs for enhanced performance. However, this performance enhancement comes with a drawback, as these power-intensive components can reduce the fuel efficiency of an AV by nearly 10% compared to a conventional vehicle [10]. Moreover, the increased energy consumption leads to higher costs, and the additional onboard sensors and components further add to the expenses of owning an AV. For instance, a typical LiDAR sensor by Velodyne, designed for AV applications, can cost up to $8k USD per unit [11]. In addition, the placement of LiDAR sensors on AVs can result in a limited perspective, limiting their capacity to fully capture the surroundings. This is contrasted by applications, such as aerial scans using LiDAR on aircraft, which are aimed at quickly covering extensive areas [12], [13].

There are relatively few studies addressing the concerns associated with using LiDAR in AVs [14], [15]. While the majority of research emphasizes leveraging LiDAR technology for enabling AV navigation [16], [17], [18], only a select few tackle the challenges of deploying LiDAR with AVs. One innovative approach, as suggested by Jayaweera et al. [19], involves mounting LiDAR sensors at elevated positions along roadways within an urban ITS framework—this configuration is known as elevated LiDAR (ELiD). ELiD units, such as roadside units (RSU) [20], are integrated with other ITS components to facilitate efficient communication with passing AVs. The intense sensing and processing tasks that AVs must perform while processing LiDAR onboard will be partially handled by the ITS and cloud infrastructure.

While ELiD is primarily envisioned for AV navigation applications, its utility extends to other potential areas that could significantly benefit growing urban centers. A particularly valuable application lies in infrastructure monitoring. With urban expansion and technological advancements, maintaining the integrity of existing infrastructure becomes crucial. ELiD, for instance, emerges as a promising approach for monitoring subsidence, a phenomenon where excessive groundwater extraction causes the ground to gradually sink [21], posing significant challenges in major cities, such as Mexico City and Beijing.

An inventive method for handling data from LiDAR sensing is ELiD. The assignment of ELiD computational tasks to servers located far away is a major issue that Jayaweera et al. [19] drew attention to in order to circumvent the limitations of local resources, which might not have the capacity to process onsite [22] or might use excessive energy [10]. This problem is new for LiDAR data handling, but it is similar to problems with mobile edge computing (MEC), which transfers data processing from mobile devices to edge servers in order to reduce battery use. Consequently, tasks demanding substantial computational power for mobile applications are often executed within the MEC framework. Given that LiDAR sensors generate huge amounts of data, which increase with enhanced scanning precision [22], and the objective of ELiD systems to reduce the computational demands on AVs [19], incorporating MEC technology into ELiD's backhaul architecture appears to be a strategic choice.

Compared to in-network processing, which consumes a lot of power, data transfer is a significant performance bottleneck in a cloud-only backend, struggling with the transmission of very large data volumes [23]. By incorporating edge computing into the backhaul, MEC task offloading is a developing field that attempts to relieve the network bottleneck that the cloud is experiencing. ELiDs connect with each other through edge computing devices, which are lower power and situated closer to the network gateway on the periphery [24], [25], [26], [27]. These edge devices may not have as much processing capability as cloud servers, but they save a lot of network transmission latency because of their close proximity to the data source. This decrease in latency compensates for their slower processing capabilities, particularly given the substantial improvements in MEC processor performance with each new generation [28].

Although MEC-based computing has advantages, it also has drawbacks. In order to effectively optimize the utilization of computational resources, MEC presents various destinations and routes for communication traffic, requiring creative solutions for backhaul routing and work distribution. Kong [29] developed an optimization problem to divide up ELiD processing jobs across virtual machines in the backhaul network with the goal of cutting costs while meeting system requirements, such as RAM availability, and meeting processing demands. Cost-effective MEC backhaul optimization has also been the subject of writing by other authors [30], [31]. Santos et al. [32] created a mixed-integer nonlinear programming (MINLP) problem to optimize wireless backhaul networks in order to lower latency. Using robust optimization techniques [33], and limiting energy consumption [34] were the two other strategies for optimizing backhaul networks. Our objective is to combine these tactics, taking care of routing and edge offloading in order to reduce total latency, which is essential for guaranteeing the security of AV operations.

The positioning of ELiDs presents another difficulty, in their brief discussion of ELiD placement, Jayaweera et al. [19] made the assumption that each ELiD would be positioned to cover a complete road grid. Because of infrastructural or resource limitations, it is essential to optimize the location of ELiDs. Optimization in ITS infrastructure planning is exemplified by literature examples, such as RSU placement [20] and UAV station positioning [35]. ELiD placement along a single roadway was examined by Lucic et al. [36]; our objective is to build on this work by taking into account a 2-D road grid and adding more flexibility to ELiD placing.

In this article, we present a comprehensive solution for integrating ELiD with MEC to enhance LiDAR-based traffic monitoring in smart cities. We first address the ELiD placement optimization, followed by tackling the ELiD backhaul routing problem to minimize latency when sending data to MEC servers. The key contributions of this article are as follows.

1) We explore the application of ELiD for traffic monitoring and as a support mechanism for AVs within ITS.
2) We formulate two optimization problems: the first problem seeks to maximize the effective coverage of an array of ELiDs across a 2-D urban road, while the second focuses on reducing the total network latency through offloading tasks to edge servers.
3) Given the critical role of LiDAR sensing in AVs for safety [37], we have crafted a fast heuristic method for real-time task allocation. These efforts cater to the requirement for a backhaul management system that efficiently directs ELiD data from sensors to servers (edge or cloud) using current network infrastructures.

Thus, in the subsequent sections of this article, we introduce a detailed ELiD planning framework tailored for urban environments. This framework accounts for the strategic positioning of ELiDs along a realistic road grid and the efficient routing of ELiD data to processing servers within a fixed-topology backhaul. A simulation-based evaluation to validate the framework's effectiveness in urban traffic monitoring and AV navigation is then performed. Simulations show that integrating ELiD with MEC significantly reduces latency and improves LiDAR data accuracy.

The rest of this article is organized as follows. Section II defines the issue space, together with the underlying assumptions and mathematical models, and sets, parameters, and indexes associated with the problem formulation. The backhaul network model is given in Section III. The problem is formalized in Section IV. This is followed by a discussion of the heuristic design in Section V. We next review the numerical results in Section VI. Finally, Section VII concludes this article.

II. ELiD Placement Settings

In this section, we study the system model of the ELiD planning framework. We first examine the model pertaining to the ELiD units' locations. The Nomenclature contains an exhaustive list of all the notations used in this article.

1) ELiD Characteristics: The maximum number of ELiDs that may be installed is calculated as $|\Lambda|$ x $|\mathcal{R}|$. This computation may take into account financial constraints. We consider a set of ELiDs $\lambda \in \Lambda$ to be installed along roadways $r \in \mathcal{R}$. Physically, every ELiD is the same, with θ and ϕ for the horizontal and vertical FoV angles, respectively, and a scan refresh rate f_{scan}, which is the frequency at which the ELiD rescans its surroundings. According to Fig. 1, the ELiDs are placed in fixed, elevated, and clear positions along the road, such as on lamp poles or the sides of buildings, with their apertures facing the road.

In addition to the aforementioned fixed attributes, the ELiDs have variable attributes associated with their coverage of the roadway. The effects of these attributes are shown in Fig. 2.

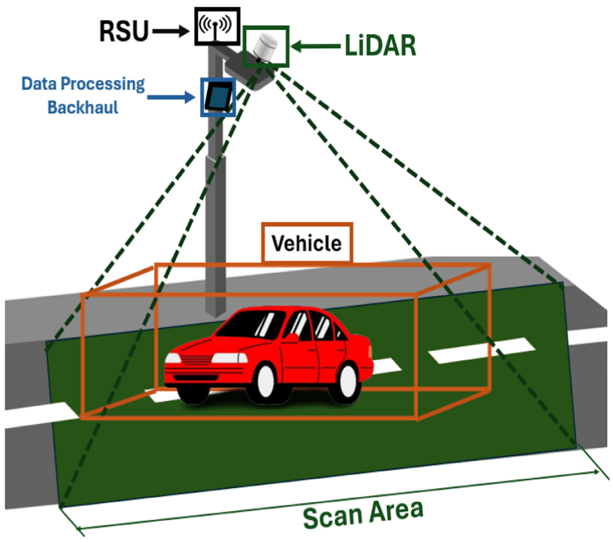


Fig. 1. Illustration of how an ELiD may cover an urban roadway.

The list includes the geographic information system (GIS) coordinates associated with the location of the ELiD λ placed along a roadway r (denoted by $(x_1^{\lambda r}, x_2^{\lambda r}, x_3^{\lambda r}) = \boldsymbol{x^{\lambda r}} \in \mathbb{R}^3$), where $x_1^{\lambda r}$ represents the latitude, $x_2^{\lambda r}$ represents the longitude, and $x_3^{\lambda r}$ represents the altitude relative to the ground of the ELiD's position. In addition, the ELiDs have a set of angles $(\rho_1^{\lambda r}, \rho_2^{\lambda r}) \in \boldsymbol{\rho^{\lambda r}}$, where $\rho_1^{\lambda r}$ represents the vertical rotation of the ELiD relative to the axis orthogonal to the ground (modeled as a flat plane), and $\rho_2^{\lambda r}$ represents the rotation relative to the direction perpendicular roadway r plane's normal vector and the direction of the traffic flow.

Since we assume that the ELiDs are mounted at fixed distance y_{min}^r away from the roadway r (which corresponds to the distance between the buildings and the curbside of the road), the following relationship must hold:

$$\rho_2^{\lambda r} = \tan^{-1}\left(\frac{y_{\text{min}}^r}{x_3^{\lambda r}}\right). \tag{1}$$

2) Roadway Characteristics: Characteristics of the road grid $\mathcal{R}$ also need to be taken into account while modeling this problem. We presume that the city has a conventional roadway grid since we are considering a dense urban region as our problem setting (see Fig. 3). $\boldsymbol{y}_{\text{start}}^{\boldsymbol{r}}, \boldsymbol{y}_{\text{end}}^{\boldsymbol{r}}, \boldsymbol{y_1^r}, \boldsymbol{y_2^r}, \boldsymbol{y_3^r}, \boldsymbol{y_4^r} \in \mathbb{R}^3$) are the six key coordinates that define the geospatial position of each road segment $r \in \mathcal{R}$. From these coordinates, we can express the road length D^r and the road's width $y_{\text{max}}^r - y_{\text{min}}^r$ as follows:

$$D^r = ||\boldsymbol{y}_{\text{start}}^{\boldsymbol{r}} - \boldsymbol{y}_{\text{end}}^{\boldsymbol{r}}||_2 \tag{2a}$$

$$y_{\text{max}}^r = ||\boldsymbol{y}_{\text{start}}^{\boldsymbol{r}} - \boldsymbol{y_3^r}||_2 \tag{2b}$$

$$y_{\text{min}}^r = ||\boldsymbol{y}_{\text{start}}^{\boldsymbol{r}} - \boldsymbol{y_1^r}||_2 \tag{2c}$$

where y_{max}^r is the distance from the side of the potential ELiD location to the far curb on the other side of the street. Since the roads in the problem space are assumed to be straightaway, the roadways can be modeled as rectangles, where the corners are determined by their GIS coordinates, and these coordinates can be utilized to calculate the road width and length. We consider the start of the roadway to be the northwestern-most point

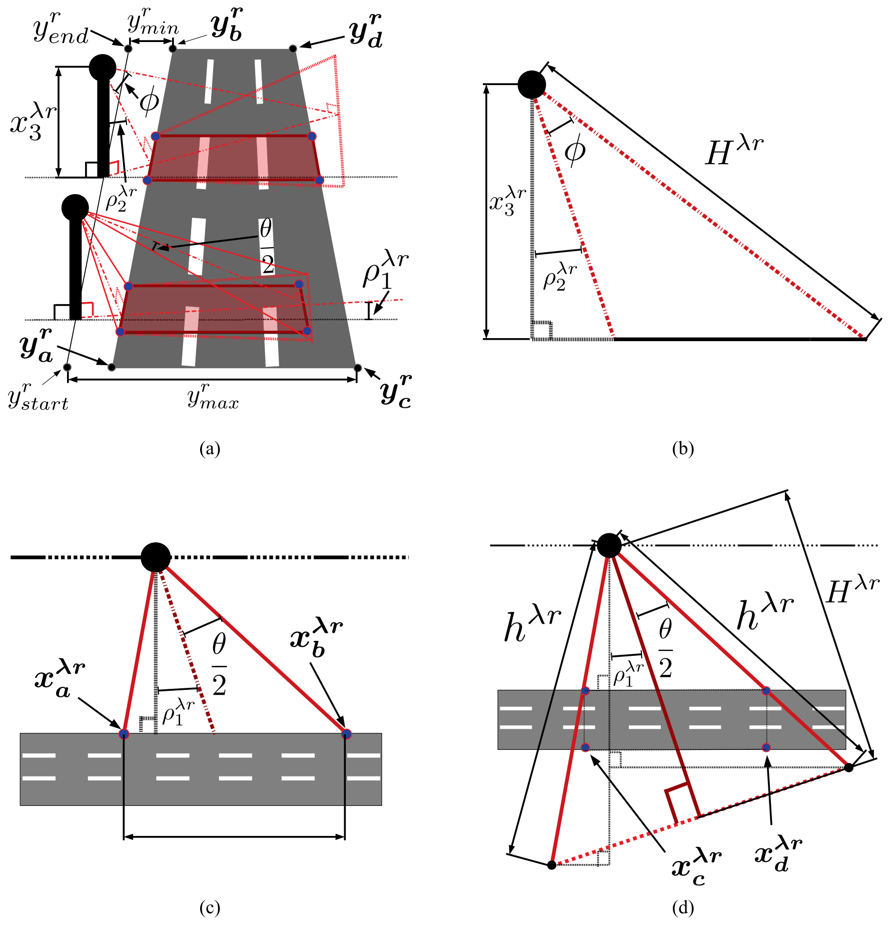


Fig. 2. Visualizing ELiD coverage. (a) Isometric view of the ELiD coverage of a roadway. (b) Cross-sectional perspective of the same ELiD, showing how the ELiD coverage spans the roadway according to the ELiD altitude $x_3^{\lambda r}$, and the horizontal FoV angle ϕ. (c) Top–down view of the ELiD, with a focus on the triangular plane that spans from the ELiD aperture to the near-side curb. This perspective shows how $\boldsymbol{x}_a^{\lambda r}$ and $\boldsymbol{x}_b^{\lambda r}$ are derived from y^r_{min}, θ, and $\boldsymbol{x}^{\lambda r}$. (d) Another top–down view, this time focusing on the triangular plane that comprises the "top" of the ELiD coverage zone. This view is used to show how $\boldsymbol{x}_c^{\lambda r}$ and $\boldsymbol{x}_d^{\lambda r}$ are derived.

associated with the road boundaries. The roadway may also be at an angle τ^r relative to a heading directly east.

We introduce a new significance measure $I^{rg} \in [0, 1]$, a continuous and normalized utility function, where a score of 0 indicates that an area does not require coverage and a value of 1 indicates a very important, busy, or risky area. This is due to the possibility that some portions $g \in \mathcal{G}$ of the road r have busier intersections. The boundaries of each section a can also be defined using GIS coordinates $\boldsymbol{i}_1^{rg}, \boldsymbol{i}_2^{rg}, \boldsymbol{i}_3^{rg}, \boldsymbol{i}_4^{rg} \in \mathbb{R}^3$.

3) Modeling ELiD Coverage: With the spatial characteristics of the roadway and ELiD considered, we now model the coverage of ELiDs λ along a roadway r. As illustrated in Fig. 2, we can approximate the coverage area on the roadway as a rectangular area with four coordinates $\boldsymbol{x}_1^{\lambda r}, \boldsymbol{x}_2^{\lambda r}, \boldsymbol{x}_3^{\lambda r}, \boldsymbol{x}_4^{\lambda r} \in \mathbb{R}^3$ that comprise the corners of the rectangle. Based on the geometric characteristics of the ELiD coverage model, these coordinates are expressed as follows:

$$\boldsymbol{x}_1^{\lambda r} = \left(x_1^\lambda - d_a^{\lambda r}, x_2^\lambda + y^r_{\text{min}}, 0\right) \tag{3a}$$

$$\boldsymbol{x}_2^{\lambda r} = \left(x_1^\lambda + d_b^{\lambda r}, x_2^\lambda + y^r_{\text{min}}, 0\right) \tag{3b}$$

$$\boldsymbol{x}_3^{\lambda r} = \left(x_1^\lambda - d_a^{\lambda r}, x_2^\lambda + y^r_{\text{min}} + L^\lambda_{\text{eff}}, 0\right) \tag{3c}$$

$$\boldsymbol{x}_4^{\lambda r} = \left(x_1^\lambda + d_b^{\lambda r}, x_2^\lambda + y^r_{\text{min}} + L^\lambda_{\text{eff}}, 0\right) \tag{3d}$$

where

$$d_a^{\lambda r} = y^r_{\text{min}} \tan\left(\frac{\theta}{2} - \rho_1^{\lambda r}\right) \tag{4a}$$

$$d_b^{\lambda r} = y^r_{\text{min}} \tan\left(\frac{\theta}{2} + \rho_1^{\lambda r}\right) \tag{4b}$$

$$L^{\lambda r}_{\text{eff}} = \min\{L^{\lambda r}_{\text{width}}, y^r_{\text{max}} - y^r_{\text{min}}\} \tag{4c}$$

$$L^{\lambda r}_{\text{width}} = h^{\lambda r} \min\left\{\cos\left(\frac{\theta}{2} - \rho_1^{\lambda r}\right), \cos\left(\frac{\theta}{2} + \rho_1^{\lambda r}\right)\right\} \tag{4d}$$

$$H^{\lambda r} = x_3^{\lambda r} \sec(\rho_2^{\lambda r} + \phi) \tag{4e}$$

$$h^{\lambda r} = H^{\lambda r} \sec\left(\frac{\theta}{2}\right). \tag{4f}$$

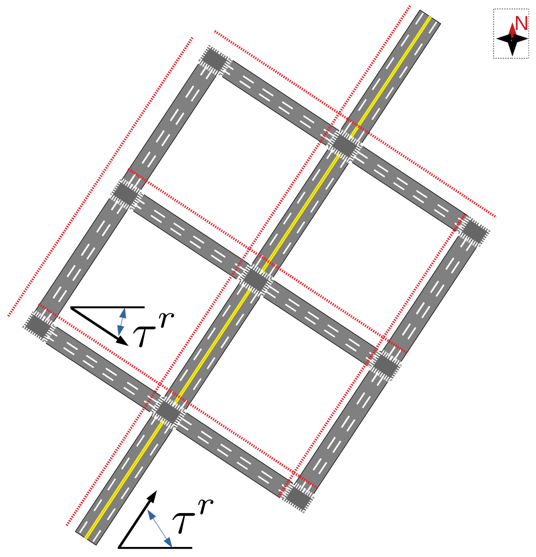


Fig. 3. Example of a street grid that needs to be covered by the ELiD system. The longest road along the middle is the main thoroughfare. The red lines represent the locations along the roadway where ELiDs may be placed. The angles of inclination/declination relative to eastbound τ^r are shown—we only have illustrated two because the angles of all of the other roads can be inferred from the ones provided.

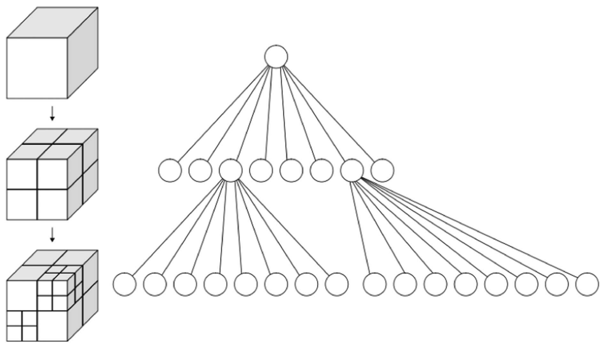


Fig. 4. Visualization of the octree structure representing the LiDAR resolution.

Based on the rectangle bounding coordinates, we can therefore express the effective area coverage as follows:

$$A^{\lambda r}_{\text{cov}} = ||\boldsymbol{x}^{\lambda r}_1 - \boldsymbol{x}^{\lambda r}_2||_2 \cdot ||\boldsymbol{x}^{\lambda r}_1 - \boldsymbol{x}^{\lambda r}_3||_2 \tag{5}$$

where $||.||_2$ is the Euclidean norm.

4) Modeling ELiD Data Production and Power Consumption: The most commonly used data structure to represent the point set cloud-based 3-D model generated by an ELiD scan of its surroundings is the octree structure [38]. The scan sector of an octree can be abstracted as a recursive set of cubes. If an object or boundary is detected in the cube, it is split into eight subcubes. In the subcube, a detection is assigned a value of 1, and a nondetection is assigned a value of 0. This process is repeated recursively until a finite depth δ is reached—this depth parameter is based on the scan resolution of the ELiD. In addition to being represented geometrically as a collection of cubes, the octree is also represented as an 8-tree, where the subcubes are child nodes of the parent cube/node, as shown in Fig. 4. Assuming a worst-case scenario of a maximally populated octree given a specific depth parameter δ, the amount of data produced per scan volume $\Gamma^{\lambda r}$ (bytes/m^3) is expressed as follows:

$$\Gamma^{\lambda r} = 8^{\delta - 2} + 12 \tag{6}$$

where the additional 12 bytes is the amount of data required to store $\boldsymbol{x}^{\lambda r}$ as a 3-tuple of 32-bit floats. We can, therefore, estimate the total amount of data produced per scan $\Delta^{\lambda r}$, which can be expressed as follows:

$$\Delta^{\lambda r} = z_{\text{cov}} \times A^{\lambda r}_{\text{cov}} \times \Gamma^{\lambda r} \times \frac{1}{f_{\text{scan}}} \tag{7}$$

where z_{cov} is a predetermined height of the coverage prism.

Since the ELiD is connected to an RSU that transmits ELiD point set map to AVs passing by, we approximate the energy consumption of the joint RSU-ELiD apparatus $E^{\lambda r}$ as follows:

$$E^{\lambda r} = E^{\lambda r}_{\text{RSU}} + E^{\lambda r}_{\text{ELiD}} + E_0 \tag{8a}$$

$$E^{\lambda r}_{\text{RSU}} = \frac{P_{\text{trans}} \times \Delta^{\lambda r}}{R_{\text{trans}}} \tag{8b}$$

$$E^{\lambda r}_{\text{ELiD}} = \frac{P^{\lambda r}}{f_{\text{scan}}} \tag{8c}$$

where $E^{\lambda r}_{\text{RSU}}$ is the RSU energy consumption, based on the power (W) P_{trans} required to transmit $\Delta^{\lambda r}$ MB of data at a rate of R_{trans} (MB/s), $E^{\lambda r}_{\text{ELiD}}$ is the amount of energy required to power the ELiD scan laser of power $P^{\lambda r}$ at a frequency of f_{scan} Hz, and E_0 is the base power consumption lost to internal component impedance.

III. Backhaul Network Model

The LiDAR sensors are linked to a backhaul network that links ELiDs to a dispersed network of computers on the edge and in the cloud, enabling fast processing of the ELiD data. We present the backhaul model for our problem scenario of relevance in this section.

A. Network Topology

The network is comprised of a set of nodes $\mathcal{V}$. These nodes consist of two classes of devices: ELiDs, denoted by $(\lambda, r) \in \Lambda \times \mathcal{R}$, and computer network nodes $c \in \mathcal{C}$, such that $\mathcal{V} = \Lambda \cup \mathcal{C}$. We can further subdivide the set of computer network nodes $\mathcal{C}$ into three subclasses: routers, MEC devices, and the cloud server. Routers serve only to act as connection hubs in the network—they do not have any processing capabilities. MEC devices are devices with relatively slow computational devices and smaller amounts of RAM, but are located much closer to the ELiDs than the cloud server. The cloud server is a powerful virtual machine instance located in a data center that is physically far away from the urban street grid defined in our problem. As a result, the expected hop-by-hop (hbh) transmission latency is much lower to an MEC device compared to the one to the cloud server.

The backhaul network topology is fixed (i.e., all of the nodes are linked by fiber-optic cables). We denote the fiber-optic links as $l_{ij} \in \mathcal{L}$, where $i, j \in \mathcal{V}$. We construct the network in a hybrid bus/star/daisy-chain topology that is a common configuration

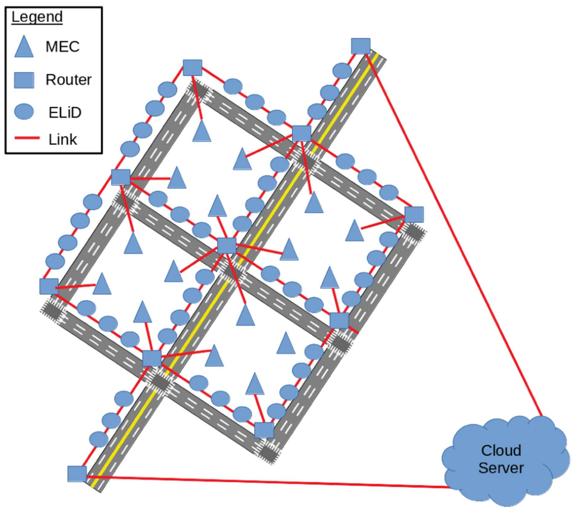


Fig. 5. Proposed backhaul network topology, along with the road grid that the communications infrastructure is installed alongside with. Routers are located at intersections, ELiDs are located along the road between routers, edge servers are located nearby and are connected to routers, and the cloud is located far away, accessible only from the main trunk lines. Note that the number of ELiDs along the roadway may vary based on the placement determined by the problem formulated in this article.

for computer networks. In this problem setting, the main bus runs along the main street of the street grid. Each intersection has a router. Routers may be connected to MEC devices nearby, and all ELiDs are positioned in a daisy chain between the nodes at the ends of their streets. At the ends of the bus topology, the two routers have direct links to the cloud server that are slow, because these links must share throughput with other Internet traffic. See Fig. 5 for an illustration of the network topology with the road grid as a backdrop. Note that other topologies of the network backhaul can be also addressed using our formulated optimization problems and heuristic algorithms models.

B. Network Characteristics

After defining the network topology, we now discuss the various characteristics that will affect how the data are routed in the backhaul network. Each ELiD λ along each roadway r produces $\Delta^{\lambda r}$ MB of data per scan; this relationship was highlighted in (7), and is dependent on the ELiD's configuration. Each link l_{ij} has a data throughput rate of T_{ij} in GB/s. The links are bidirectional and can handle an arbitrary number of signals passing through, as each can be broadcast on a slightly different visible light spectra. Along each link, the direction of signals t may be associated with uplink messages from an ELiD (λ, r) to a server c, or downlink messages from a server to an ELiD. We, therefore, denote the set of directions as $\mathcal{T} = \{\text{up,down}\} : t \in \mathcal{T}$.

We assume that, after the server processes the data from a LiDAR (λ, r), the processed map returned to the LiDAR to be transmitted back to the passing AVs is some fraction $\gamma < 1$ of the initial map, as extraneous information may have been filtered out. As a result, the size of the processed map $\Delta_c^{\lambda r}$ in MB is expressed as follows:

TABLE I
Decision Variables for $(\mathcal{P})$

Variable	Type	Definition
$\boldsymbol{\epsilon}^{\lambda r}$	Continuous $(\mathbb{R}^3)$	ELiD Position
$\boldsymbol{\rho}^{\lambda r}$	Continuous $(\mathbb{R}^2)$	ELiD Rotation
$\sigma^{\lambda r}$	Binary $(\mathbb{Z})$	Placement of ELiD λ along road r

$$\Delta_c^{\lambda r} = \gamma \times \Delta^{\lambda r}. \tag{9}$$

Each server in the network c has M_c GB of onboard RAM—the total amount of RAM offered across all of the computers in the network limits how much data may be processed. Routers contain no onboard RAM, as they only route messages. MECs have small amounts of onboard RAM, and the cloud server offers an abundance of RAM. In order to estimate the speed in which servers process data, we can estimate the server processing throughput α^c based on the image processing benchmarks of the server's processing units, as LiDAR data are similar to image data if reorganized into voxel grids (3-D arrays). Since the ELiD data may be processed in a similar manner to image data, we assume that the data are embarrassingly parallel (i.e., exploiting computational parallelism to accelerate processing is trivial, and resources scale linearly—doubling the processing power halves the processing speed and doubling the number of jobs halves the amount of available resources).

IV. Problem Formulation

In this section, we formulate two mathematical programming problems. One is an MINLP problem that aims to address the placement of ELiD units, and the other is a quadratic integer programming (QIP) problem that is a variant of the quadratic assignment problem, which aims to route the ELiD data through the backhaul network. Note that outcomes of the first optimization problem (i.e., ELiD placement) may impact the outcomes of the second problem (ELiD data backhaul routing).

A. Optimization Problem for ELiD Planning

Our main goal in this problem (which we will refer to as the problem $(\mathcal{P})$ for the rest of this article) is to maximize the effective coverage area of the ELiDs that have been put on the urban road grid. Table I defines the decision variables that are necessary to formulate the problem.

1) Coordinate Transformation: As mentioned in Section II-A2, each roadway has a length of D^r meters, a width of $y^r_{\text{max}} - y^r_{\text{min}}$, and can be defined by a set of coordinates that comprise the corners of the roadway. In order to standardize ELiD placement along each roadway, independent of the orientation or direction of the roadway, we develop a coordinate transformation, where the $\boldsymbol{y}^r_{\text{start}} \rightarrow \boldsymbol{m}^r_{\text{start}} = (0,0,0)$ is the origin of the transformed coordinate plane, $\boldsymbol{y}^r_{\text{end}} \rightarrow \boldsymbol{m}^r_{\text{end}} = (D^r,0,0)$, and the road is oriented to be parallel to the east–west axis. This transformed perspective can be represented in Fig. 2(c) and (d).

There are two key steps in the transformation: translation and rotation. In order to translate a coordinate $\boldsymbol{y}^r \rightarrow \boldsymbol{m}^r : \boldsymbol{m}^r_{\text{start}} =$

$(0,0,0)$, we must define the transformed coordinates $\boldsymbol{m}^{\boldsymbol{r}}$ as follows:

$$\boldsymbol{m}^{\boldsymbol{r}} = \boldsymbol{R}_{\text{coord}}(-\tau^r)[\boldsymbol{y}^{\boldsymbol{r}} - \boldsymbol{y}^{\boldsymbol{r}}_{\text{start}}] \tag{10}$$

where $\boldsymbol{R}_{\text{coord}}(-\tau^r)$ is the rotation matrix that rotates a coordinate $-\tau_r$ radians about the axis orthogonal to the ground, and $\boldsymbol{m}^{\boldsymbol{r}}, \boldsymbol{y}^{\boldsymbol{r}}$, and $\boldsymbol{y}^{\boldsymbol{r}}_{\text{start}}$ are $\mathbb{R}^3$ column vectors. The rotation matrix $\boldsymbol{R}_{\text{coord}}(-\tau^r)$ is expressed as follows:

$$\boldsymbol{R}_{\text{coord}}(-\tau^r) = \begin{bmatrix} \cos(-\tau^r) & -\sin(-\tau^r) & 0 \\ \sin(-\tau^r) & \cos(-\tau^r) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

Following the prior definitions, we define a decision variable $\boldsymbol{\epsilon}^{\boldsymbol{\lambda r}}$, the transformed coordinate of the ELiD position, which is expressed as follows:

$$\boldsymbol{\epsilon}^{\boldsymbol{\lambda r}} = \boldsymbol{R}_{\text{coord}}(-\tau^r)[\boldsymbol{x}^{\boldsymbol{\lambda r}} - \boldsymbol{y}^{\boldsymbol{r}}_{\text{start}}]. \tag{12}$$

The ELiD's actual position may be reconstructed from $\boldsymbol{\epsilon}^{\boldsymbol{\lambda r}}$, τ^r, and $\boldsymbol{y}^{\boldsymbol{r}}_{\text{start}}$, which are decomposed characteristics of the coordinate, as follows:

$$\boldsymbol{x}^{\boldsymbol{\lambda r}} = [\boldsymbol{R}_{\text{coord}}(\tau^r)\boldsymbol{\epsilon}^{\boldsymbol{\lambda r}}] + \boldsymbol{y}^{\boldsymbol{r}}_{\text{start}} \tag{13}$$

since $\boldsymbol{R}_{\text{coord}}(\tau^r) = \boldsymbol{R}_{\text{coord}}{}^{-1}(-\tau^r)$.

2) Objective Function: We aim to configure the placement of ELiDs so as to maximize their effective coverage of the road grid and reduce the coverage overlap between adjacent ELiDs—an area along a roadway r should only be counted as covered once if it is covered by two ELiDs.

An ELiD λ was formerly specified in Section II-A1 to have a coverage area shaped like a rectangle, determined by four points based on the geometric relationships as a result of the ELiD's position and rotation. If this ELiD λ were to be placed along a roadway r, we can transform the relevant GIS coordinates associated with λ to the standardized coordinates $\boldsymbol{\epsilon}^{\boldsymbol{\lambda r}}$ via (12). Since all ELiDs must be placed y^r_{min} meters away from the roadway, $\epsilon^r_{\lambda 2} = 0\ \forall \lambda, r$. Therefore, when dealing with the standardized coordinates, we only need to be concerned with the "east–west" position and the altitude regarding placed ELiDs. Therefore, we deduce that $\epsilon^{\lambda r}_1 \in [0, D^r]$ and $\epsilon^{\lambda r}_3 \in [z_{\text{min}}, z_{\text{max}}]$.

For each ELiD λ placed along a road r, we can determine the start and end boundaries of the coverage rectangle ($\epsilon^{\lambda r}_s$ and $\epsilon^{\lambda r}_e$, respectively) by transforming $\boldsymbol{x}^{\boldsymbol{\lambda r}}_{\boldsymbol{a}} \to \boldsymbol{\epsilon}^{\boldsymbol{\lambda r}}_{\boldsymbol{a}}$ and $\boldsymbol{x}^{\boldsymbol{\lambda r}}_{\boldsymbol{b}} \to \boldsymbol{\epsilon}^{\boldsymbol{\lambda r}}_{\boldsymbol{b}}$ via (12). The decision variables $\epsilon^{\lambda r}_s$ and $\epsilon^{\lambda r}_e$ can therefore be expressed as follows:

$$\epsilon^{\lambda r}_s = \max(\epsilon^{\lambda r}_{a1}, 0) \tag{14a}$$

$$\epsilon^{\lambda r}_e = \min(\epsilon^{\lambda r}_{b1}, D^r). \tag{14b}$$

The max/min restrictions are put into place to ignore coverage outside of the roadway, as these do not make the coverage more effective in this problem setting.

The collection of these $\epsilon^{\lambda r}_s$ and $\epsilon^{\lambda r}_e$ values can be organized into a set of boundaries $\mathcal{B}^r$, along with the road boundaries 0 and D^r. The collection of these boundaries are sorted in ascending order (i.e., 0 to D^r, and can be denoted as $\zeta^r_\beta \in \mathcal{B}^r$). These boundaries are used to divide the entire road r coverage into subrectangles, which have an area $\mathcal{A}^r_\beta$, which can be expressed as follows:

$$\mathcal{A}^r_\beta = L^r_\beta \times W^r_\beta \tag{15a}$$

$$L^r_\beta = \zeta^r_\beta - \zeta^r_{\beta-1} \tag{15b}$$

$$W^r_\beta = \max_{\lambda \in \Lambda}\{\eta^{\lambda r}_\beta \times \sigma^{\lambda r} \times ||\boldsymbol{x}^{\boldsymbol{\lambda r}}_{\boldsymbol{a}} - \boldsymbol{x}^{\boldsymbol{\lambda r}}_{\boldsymbol{c}}||_2\} \tag{15c}$$

where $\eta^{\lambda r}_\beta$ is a binary indicator that equals 1 if an ELiD λ covers a subrectangle β of the roadway r.

After defining the areas for the subrectangles, we must consider the weighted coverage based on the importance scores I^{rg}. First, we utilize (12) to gain the following transformed coordinates of the bounding box defining the importance score coverage: $\boldsymbol{i}^{\boldsymbol{rg}}_{\boldsymbol{1}} \to \boldsymbol{\iota}^{\boldsymbol{rg}}_{\boldsymbol{1}}$ and $\boldsymbol{i}^{\boldsymbol{rg}}_{\boldsymbol{2}} \to \boldsymbol{\iota}^{\boldsymbol{rg}}_{\boldsymbol{2}}$. Next, we utilize the importance score boundaries along with the coverage boundaries to consider the likely possibility that an ELiD may cover two different importance zones. Thus, the weighted importance score for a subrectangle π^r_j can be expressed as follows:

$$\pi^r_\beta = \begin{cases} \frac{1}{L^r_\beta}\sum_g \Omega^{rg}_\beta, & \text{if } L^r_\beta > 0 \\ 0, & \text{otherwise} \end{cases} \tag{16a}$$

$$\Omega^{rg}_\beta = \nu^{rg}_\beta I^{rg}(\min(\zeta_\beta, \iota^{rg}_{11}) - \max(\zeta_{\beta-1}, \iota^{r(g-1)}_{11})) \tag{16b}$$

where ν^{rg}_β is a binary indicator variable that equals 1 if the relevance sector g is located within the rectangle j along roadway r.

With respect to the previously derived quantities, we can define the model objective function as the effective area coverage score for the entire roadway grid A_{grid} as follows:

$$A_{\text{grid}} = \frac{\sum_{r\in\mathcal{R}}\sum_{\beta\in\{2\ldots|\mathcal{B}^r|\}}\mathcal{A}^r_\beta\pi^r_\beta}{\xi\sum_{r\in\mathcal{R}}D^r(y^r_{\text{max}} - y^r_{\text{min}})} \tag{17}$$

where ξ is the predefined percentage of the grid that should be covered.

3) Constraints: There are physical limitations of the system that must be codified as constraints.

a) Data Capacity Constraint: ELiD placement cannot exceed the total RAM contained in the backhaul network. This constraint can be expressed as follows:

$$\frac{\sum_{\lambda\in\Lambda}\sum_{r\in\mathcal{R}}\sigma^{\lambda r}\Delta^{\lambda r}}{\sum_{c\in\mathcal{C}}M_c} - 1 \le 0. \tag{18}$$

b) Energy Capacity Constraint: ELiDs should not exceed their maximum safety ratings for operating energy consumption (E_{max}) as follows:

$$\frac{\sum_{r\in\mathcal{R}}\sigma^{\lambda r}E^{\lambda r}}{E_{\text{max}}} - 1 \le 0 \quad \forall\lambda\in\Lambda. \tag{19}$$

c) Orientation Constraints: The vertical rotation of the ELiD is dependent on the vertical position of the ELiD, because the lowest coverage plane must intersect with the near-side curb as indicated by (1).

*a) Decision Variable Boundaries:*The following restrictions are placed on the decision variables based on the system model and problem definitions. They define the limits of the different

TABLE II
DECISION VARIABLES FOR $(\mathcal{Q})$

Variable	Type	Definition
$\mu_{ij}^{\lambda rt}$	Binary $(\mathbb{Z})$	Transmission in t direction from/to (λ, r) is sent along a link i, j
$J_c^{\lambda r}$	Binary $(\mathbb{Z})$	(λ, r) is assigned to server c

primary and auxiliary decision variables

$$\epsilon_1^{\lambda r} \in [0, D_r] \quad \forall \lambda, r \in \Lambda, \mathcal{R} \tag{20a}$$

$$\epsilon_2^{\lambda r} = 0 \quad \forall \lambda, r \in \Lambda, \mathcal{R} \tag{20b}$$

$$\epsilon_3^{\lambda r} \in [z_{\min}, z_{\max}] \quad \forall \lambda, r \in \Lambda, \mathcal{R} \tag{20c}$$

$$\rho_2^{\lambda r} \in \left(-\left(\frac{\pi}{2} - \frac{\theta}{2}\right), \left(\frac{\pi}{2} - \frac{\theta}{2}\right) \right) \quad \forall \lambda, r \in \Lambda, \mathcal{R} \tag{20d}$$

$$\sigma^{\lambda r} \in \{0, 1\} \quad \forall \lambda, r \in \Lambda, \mathcal{R}. \tag{20e}$$

4) Overlap Penalty Regularization: When placing the ELiD units, we aim to minimize excessive overlap when covering the roadway. Recall, in the objective function definition, we defined an indicator variable, $\eta_\beta^{\lambda r}$, that equals 1 if an ELiD covers a subsection. We can thus define the number of ELiDs covering each subsection (o_β^r) as follows:

$$o_\beta^r = \sum_{\lambda \in \Lambda} \eta_\beta^{\lambda r}. \tag{21}$$

From this, we can define an ELiD λ as having its coverage completely dominated if it overlapped in every section it covers. We mathematically encode this as follows:

$$\chi^{\lambda r} = \begin{cases} 1, & \text{if } o_\beta^r \eta_\beta^{\lambda r} \neq 1 \quad \forall \beta \in \mathcal{B}^r \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

where $\chi^{\lambda r} = 1$ means that the ELiD is dominated.

5) MINLP Formulation: With the problem space defined, we now formulate the MINLP $(\mathcal{P})$ as follows:

$$(\mathcal{P}) \min \; -A_{\text{grid}} + \kappa \sum_{\lambda \in \Lambda} \sum_{r \in \mathcal{R}} \chi^{\lambda r}$$
$$\text{s.t. Constraints } (1), (18)\text{–}(20)$$

where κ is a regularization parameter that penalizes excess placement of ELiDs. In other words, dominated ELiDs do not contribute any additional coverage whatsoever.

B. Optimization Problem for ELiD Data Backhaul Routing

In this problem (which we will refer to as the problem $(\mathcal{Q})$), our primary objective is to minimize the total backhaul network hbh and processing latency. The decision variables that the problem formulation depends on are defined in Table II.

1) Objective Function: The aim of $(\mathcal{Q})$ is to minimize the total latency associated with the backhaul network connecting the ELiD's $(\lambda, r) \in \Lambda \times \mathcal{R}$ to the servers $c \in \mathcal{C}$. For each ELiD, we must model the total latency $S^{\lambda r}$ as a sum of the hbh $(S_{\text{hbh}}^{\lambda r})$ and processing $(S_{\text{proc}}^{\lambda r})$ latencies associated with the route taken in the backhaul network. We express these latencies for each pair ELiD denoted by λ, r as follows:

$$S^{\lambda r} = S_{\text{hbh}}^{\lambda r} + S_{\text{proc}}^{\lambda r} \tag{23a}$$

$$S_{\text{hbh}}^{\lambda r} = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \left[\mu_{ij}^{\lambda rt} \Delta^{\lambda r} \frac{n^t + (1 - n^t)\gamma}{\nu_{ij} T_{ij}} \right] \tag{23b}$$

$$S_{\text{proc}}^{\lambda r} = \sum_{c \in \mathcal{C}} \left(J_c^{\lambda r} \left[\sum_{\lambda \in \Lambda} J_c^{\lambda r} \right] \frac{\Delta^{\lambda r}}{\alpha_c} \right) \tag{23c}$$

$$\frac{1}{\nu_{ij}} = \sum_{t \in \mathcal{T}} \sum_{\lambda \in \Lambda} \mu_{ij}^{\lambda rt}. \tag{23d}$$

In (23), n^t is a binary indicator that equals 1 if $t = \text{up}$ and 0 if $t = \text{down}$. From these definitions, we express the objective function S_{total} as follows:

$$S_{\text{total}} = \sum_{r \in \mathcal{R}} \sum_{\lambda \in \Lambda} S^{\lambda r}. \tag{24}$$

2) Constraints: a) Flow Balance Constraints: All messages entering a server c should either move on to another link or should be processed in that server

$$\left[\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{V}} \mu_{ic}^{\lambda rt} \right] + (1 - n^t) J_c^{\lambda r} = \left[\sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{V}} \mu_{cj}^{\lambda rt} \right] + n^t J_c^{\lambda r}$$
$$\forall c \in \mathcal{C} \; \forall r \in \mathcal{R} \; \forall \lambda \in \Lambda. \tag{25}$$

In addition, all ELiD's (λ, r) must send out data for processing

$$\sum_{j \in \mathcal{V}} \mu_{(i=(\lambda,r))j}^{\lambda r(t=\text{up})} = 1 \quad \forall r \in \mathcal{R} \; \forall \lambda \in \Lambda \tag{26}$$

and must receive a processed map

$$\sum_{i \in \mathcal{V}} \mu_{i(j=(\lambda,r))}^{\lambda r(t=\text{down})} = 1 \quad \forall r \in \mathcal{R} \; \forall \lambda \in \Lambda. \tag{27}$$

b) Topology Constraints: All messages sent across the backhaul network must be transmitted along existing fiber-optic links—the network has a fixed topology:

$$\mu_{ij}^{\lambda rt} \leq l_{ij} \quad \forall i, j \in \mathcal{V} \; \forall \lambda \in \Lambda \; \forall r \in \mathcal{R} \; \forall c \in \mathcal{C}. \tag{28}$$

c) Direction Double-Counting Prevention: If a message to/from ELiD (λ, r) passes along a link l_{ij}, it is logically equivalent to that message passing along link l_{ji}. Therefore, we express this as follows:

$$\mu_{ij}^{\lambda rt} + \mu_{ji}^{\lambda rt} \leq 1 \quad \forall i, j \in \mathcal{V} \; \forall \lambda \in \Lambda \; \forall r \in \mathcal{R} \; \forall c \in \mathcal{C}. \tag{29}$$

d) Server RAM Capacity Constraints: All job processing is performed in memory, as disk input/output is too slow for this system to operate in a timely manner. As a result, the number of jobs running on a server c is restricted by the onboard RAM M_c

$$\sum_{r \in \mathcal{R}} \sum_{\lambda \in \Lambda} \Delta^{\lambda r} J_c^{\lambda r} \leq M_c \quad \forall c \in \mathcal{C}. \tag{30}$$

e) Job Assignment Requirements: All ELiD's (λ, r) must be assigned to a server c to have its data processed

$$\sum_{c \in \mathcal{C}} J_c^{\lambda r} = 1 \quad \forall r \in \mathcal{R} \; \forall \lambda \in \Lambda. \tag{31}$$

3) QIP Formulation: With the objective and constraints for (Q) defined, we formulate the QIP as follows:

$$(\mathcal{Q})\ \min\ S_{\text{total}}$$
$$\text{s.t. Constraints (25)–(31).}$$

The optimization problem $(\mathcal{Q})$ is formulated as a QIP problem, which is inherently NP-hard due to its quadratic objective function and integer-based decision variables. Specifically, $(\mathcal{Q})$ includes discrete routing and server assignment decisions with nonlinear latency objectives, introducing complex, interdependent constraints. Routing paths increase exponentially with network size, creating an exhaustive set of possibilities for determining efficient paths. In addition, server assignment introduces combinatorial scheduling complexity, mirroring NP-hard allocation problems. Flow balance and server memory constraints add further layers of resource allocation challenges. These combined elements lead to a computationally intensive problem characteristic of QIP, similar to NP-hard problems, such as the quadratic assignment problem and binary quadratic programming. Solving $(\mathcal{Q})$ demands advanced algorithms, particularly as network size scales, underscoring the challenge of achieving optimal solutions [39], [40].

Hence, the optimization problem cannot be optimally solved due to the nonlinearity of its constraints and objective function since $(\mathcal{P})$ contains nonlinear functions within the objective function $-A_{\text{grid}} + \kappa \sum_{\lambda\in\Lambda}\sum_{r\in\mathcal{R}} \chi^{\lambda r}$, along with the constraint defined in (1). Therefore, applying exact methods, such as branch-and-bound [41], that are implemented in off-the-shelf optimization software, such as Gurobi, will be computationally expensive.[1] In fact, the running time grows exponentially with the network size, which makes very complex to achieve an optimal solution for large-scale problems in a real-time manner [42], [43]. Therefore, in the following section, we propose to design two low complexity algorithms to solve these NP-hard problems.

V. Heuristic Design for ELiD Planning and Routing Frameworks

In this section, we design the heuristic approaches that will be applied to $(\mathcal{P})$ and $(\mathcal{Q})$, respectively.

A. Particle Swarm Optimization (PSO) for ELiD Placement

Practically, the planning problem to permanently deploy ELiDs needs to be solved once for each geographical region. To this end, we consider the use of PSO [44], a nature-inspired algorithm, which has been applied to numerous other problems with effective results [35].

Since PSO was developed initially for unconstrained search spaces, we must redefine $\mathcal{P}$ in a way that it is no longer constrained. We accomplish this by converting the constraints to exterior penalty functions that may be used as additional regularization functions for the objective. Recall that $\mathcal{P}$ was formulated in such a way that all of the model constraints can be generalized as follows:

$$h_\upsilon(\boldsymbol{X}) \le 0 \quad \forall \upsilon \in \Upsilon \tag{32}$$

where $\boldsymbol{X}$ is a collection of all of the decision variables in $\mathcal{P}$, and Υ is the set of constraints in $\mathcal{P}$. We can also express the model objective function as follows:

$$f(\boldsymbol{X}) = -A_{\text{grid}} + \kappa \sum_{\lambda\in\Lambda}\sum_{r\in\mathcal{R}} \chi^{\lambda r}. \tag{33}$$

Therefore, we can now redefine $(\mathcal{P})$ as follows:

$$(\mathcal{P}_{\text{pso}})\ \min\ f(\boldsymbol{X}) + \sum_{\upsilon\in\Upsilon} \kappa_\upsilon h_\upsilon(\boldsymbol{X}) \tag{34}$$

where κ_υ is the regularization parameter for each exterior penalty function.

With $(\mathcal{P}_{\text{pso}})$ defined as an unconstrained nonlinear mixed-integer minimization problem, we must reconcile the discrete decision variables $\sigma_{\lambda r}$ with the fact that classic PSO is designed to handle continuous decision variables. To address this shortcoming, binary PSO (BPSO) was developed [45].

In regular PSO, when a continuous variable y is updated, the following update rules are used (note, these would be applied to the decision variables $\boldsymbol{\epsilon}^{\lambda r}$ and $\boldsymbol{\rho}^{\lambda r}$):

$$v^d_{s(n+1)} = \omega v^d_{sn} + w_p r_p (p^d_{sn} - y^d_{sn}) + w_g r_g (g^d_n - y^d_n) \tag{35a}$$
$$y^d_{s(n+1)} = y^d_{sn} + v^d_{s(n+1)} \tag{35b}$$

where n is the current iteration, s is the particle index (of $|\mathcal{S}|$ particles), d is the decision variable dimension, v^d_{sn} is the particle velocity, ω is the inertia parameter, p^d_{sn} and g^d_n are the values that correspond to the local and global best values (based on an evaluation of the cost function) for the particles, respectively, w_p and w_g are the parameters that tune the particles to converge faster toward the local and global best, respectively, and r_p and r_g are random values drawn from a $U(0,1)$ distribution. We must utilize the BPSO update rules in order to take into account for $\sigma^{\lambda r}$. After updating its velocity via (35b), we determine the new position for $\sigma^{\lambda r}_{s(n+1)}$ as follows:

$$e^{\lambda r}_{s(n+1)} = \frac{1}{1 + \exp\left(v^{\lambda r}_{s(n+1)}\right)} \tag{36a}$$

$$\sigma^{rn+1}_{s\lambda} = \begin{cases} 1, & \text{if } r_e < e^{\lambda r}_{s(n+1)} \\ 0, & \text{otherwise} \end{cases} \tag{36b}$$

where r_e is a random value drawn from a $U(0,1)$ distribution. With the update rules defined, we develop our PSO procedure to terminate upon the satisfaction of either of the two following conditions: 1) the solver reaches the maximum number of iterations $n_{\max}$, or 2) there is no improvement of the fitness function (34) larger than η_{stop} for n_{stall} consecutive iterations.

B. Heuristic Algorithm for Network Backhaul Routing

We can now improve the LiDAR data routing across the backhaul network while taking into account the uplink/downlink hop-by-hop and data processing latencies after positioning the

[1] Gurobi Optimization, Inc., Gurobi Solver 8.01. [Software], Houston, TX, USA, 2018.

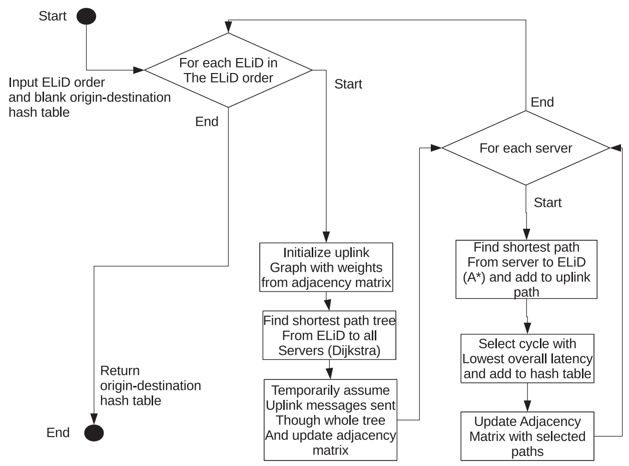


Fig. 6. Flowchart diagram for the procedure outlined in Algorithm 1.

Algorithm 1: Routing Procedure.

```
 1: function RoutingHeuristic$\Lambda_{sort}, P$
 2:   for $\lambda \in \Lambda_{sort}$ do
 3:     $G_{\text{up}} \leftarrow$ Initialize uplink directed graph with
        latency edge weights for ELiD $(\lambda, r)$
 4:     $P_{tmp}, T_{tmp} \leftarrow$ Get shortest path tree from $(\lambda, r)$
        to all dummy nodes $c'$ via Dijkstra's algorithm
        applied to $G_{\text{up}}$
 5:     for $c \in \mathcal{C}$ do
 6:       if $c$ is not a router then
 7:         $M_{tmp} \leftarrow$ update $M$ matrix temporarily with
            the current path between $(\lambda, r)$ and $c$.
 8:         $G_{\text{down}} \leftarrow$ Initialize the downlink directed
            graph with updated latency edge weights by
            $M_{tmp}$
 9:         $p_{\text{down}}, t_{\text{down}} \leftarrow$ Compute A$*$ shortest path
            between $c$ and $(\lambda, r)$ in $G_{\text{down}}$
10:         $P_{tmp,c} \leftarrow P_{tmp,c} + p_{\text{down}}$
11:         $T_{tmp,c} \leftarrow T_{tmp,c} + t_{\text{down}}$
12:     $p^* \leftarrow P_{tmp,c}$, where $c : T_{tmp,c} = min(T_{tmp})$
13:     $P_c^{\lambda r} \leftarrow p^*$
14:     Update $M$ with $P_c^{\lambda r}$
15:   return $P$
```

ELiDs in their optimum locations. In order to efficiently route ELiD data flow to MEC devices and the cloud, we devise a quick, low-complexity heuristic in this section. There are two major components to the development process. To link an ELiD (λ, r) to a server s, we first create a greedy routing algorithm for all $(\lambda, r) \in \Lambda \times \mathcal{R}$. Next, we formulate a heuristic method that establishes the proper sequence for the routing heuristic method. Algorithm 1, with the flow diagram given in Fig. 6, contains the suggested algorithm's pseudocode.

a) Initialization Phase: First, the procedure initializes an empty hash table object (see Line 2). A hash table object can be represented as a set of key/value pairs. In this setting, the hash table P represents the selected path from ELiD (λ, r) to servers c for ELiD map treatment, back to ELiD (λ, r), where the treated map is transmitted to an AV. The key is a tuple $((\lambda, r), c)$, and the value is the hbh uplink/downlink cycle that minimizes hbh and processing latency for that trip.

After initializing P, the procedure determines the minimum cycle for each ELiD (λ, r) in a greedy manner (see Line 3). Since solving for the minimal cycle that contains our desired server in a directed graph is very hard (i.e., a famous vehicle routing problem [46]), we break the problem into two subproblems: uplink/processing and downlink phases.

b) Uplink Phase: For the uplink/processing phase, we first define a directed graph where the edge weights are the expected latency, and also factor in the processing latency. First, we define a set of dummy nodes $c' \in \mathcal{C}'$, where $\mathcal{C}'$ is mapped one-to-one to the server nodes $\mathcal{C}$ (note that this one-to-one relationship is only for MEC and cloud nodes, and not the routers). The edge weights between $\mathcal{C}$ and $\mathcal{C}'$ are the expected processing rates (in MB/s) for one job in the server based on benchmark tests on the hardware. With this initial set of definitions, we define an adjacency matrix R, such that each element $T_{ij} \in T$ represents the data processing rates (if server to dummy node), or transmission capacities (if node-to-node communication link) in MB/s. Next, we define a matrix M, where each element M_{ij} represents the number of messages transmitted along a link between i and j, or the number of jobs being processed in a server c if $i = c$ and $j = c'$, if the link contained a new message from or to ELiD λ. For example, if we were initializing M from scratch, $M_{ij} = 1 \forall i, j \in \mathcal{V} \cup \mathcal{C}'$. After, we generate the adjacency matrix Γ_{up}, where $k_{ij}^{\text{up}} \in \Gamma_{\text{up}}$ represents the transmission or processing latencies between nodes i and j in the uplink graph. We determine these values by $k_{ij}^{\text{up}} = \frac{\Delta_\lambda M_{ij}}{T_{ij}}$. Γ_{up} is then used to generate the uplink graph G_{up} (see Line 4). After initializing the uplink graph's edge weights with the expected latencies Γ_{up}, we then determine the shortest path tree from ELiD λ to all mirror nodes $c' \in \mathcal{C}'$. This can be used to update two more dictionaries P_{tmp} and T_{tmp}. The keys in both are the servers $c \in \mathcal{C}$ that data from ELiD (λ, r) may be transmitted to (see Line 5). The values that the keys point to in P_{tmp} are the shortest paths between c and (λ, r), and the values in T_{tmp} are the total uplink and processing latencies for their respective paths. Now that we have the potential uplink paths, we can focus on the downlink phase of the problem.

c) Downlink Phase: In the downlink phase, we determine the shortest latency path back from the server c to ELiD (λ, r), for each server $c \in \mathcal{C}$ (see Line 6). In a similar vein to how we produced the uplink adjacency matrix Γ_{up}, we must also produce a downlink adjacency matrix Γ_{down}, where $k_{ij}^{\text{down}} \in \Gamma_{\text{down}}$ represents the downlink transmission latency. First, we must temporarily update the M matrix with the selected path $P_{\text{tmp},c} \in P_{\text{tmp}}$, and save this temporary update as M_{tmp}. For example, if we are still solving the first step of the problem, where $M_{ij} = 1 \; \forall i, j$, we first set $M_{\text{tmp}} = M$. After, for each link between two nodes i and j in the path, we do the following: $M_{\text{tmp},ij} = M_{\text{tmp},ij} + 1$ and $M_{\text{tmp},ji} = M_{\text{tmp},ji} + 1$. This is done when both an uplink and downlink are going along the same edges. Note that we can take the adaptive bandwidth sharing into consideration when estimating the hbh latencies (see Line 7). From there, we then calculate the estimated downlink latencies: $k_{ij}^{\text{down}} = \frac{\gamma \Delta^{\lambda r} M_{\text{tmp},ij}}{T_{ij}}$, where γ is the ratio of the processed ELiD map size in MB to the uplink ELiD map data size in

TABLE III
LIST OF SIMULATION MODEL PARAMETERS

Parameter	Value	Unit	Parameter	Value	Unit
Overlap regularization	0.5	N\A	F_{scan}	30	Hz
Server's Ram	250	MB/s	Z_{cov}	5	m
Max energy	100	W	δ	10	N/A
θ	1.0472	rad	R_{trans}	1	GB/s
ϕ	0.6109	rad	P_{trans}	0.1	W

TABLE IV
LIST OF SIMULATION ROAD CHARACTERISTICS

Road No	$y_{\min}$	$y_{\max}$	τ^r	D^r
1	7	19	0.3491	200
2	5	15	0.3491	110
3	5	15	0.3491	110
4	5	15	-1.2217	100
5	5	15	-1.2217	100
6	5	15	-1.2217	100

MB. Γ_{down} is then used to generate the downlink graph G_{down} (see Line 8).

After the initialization steps for the downlink phase, we then use the A$*$ algorithm to determine the shortest path p_{down} between c and (λ, r), along with the total hbh latency of this path t_{down} (see Line 9). The use of A$*$ in this scenario is motivated by the performance advantage of single-origin to single-destination shortest path calculation [47]. We then append the shortest path to the uplink shortest path to get a completed cycle (see Line 10), and we add the downlink hbh latency to the already-calculated uplink hbh and processing latency (see Line 11).

After repeating the downlink step for all servers c, we then select the cycle with the lowest total latency, denoted by p^* (see Line 12). We store the cycle p^* in P, indexed as the following key/value pair: $((\lambda, r), c) : p^*$ (see Line 13). After this, we update M with this cycle (see Line 14). The process is then repeated for each ELiD (λ, r).

The routing procedure alone is not sufficient to minimizing the total backhaul latency. This is because the order in which the ELiDs are iteratively assigned to compute resources in the backhaul system has a significant impact on the total latency, due to the fact that it is a greedy approach. In the next section, we aim to layer a metaheuristic approach around the routing procedure that aims to address shortcomings in Algorithm 1.

VI. Selected Numerical Results

In this section, we model how the suggested ELiD planning techniques would function in an urban road grid. After developing an initial run solution, we offer a sensitivity analysis for the scan depth parameter δ, as it affects the constraints for $(\mathcal{P})$. We employ the locations and road grid as a basis setting for a simulated network backhaul architecture, which is then used to evaluate the backhaul routing solvers' performance, after assessing $(\mathcal{P})$. Table III provides a summary of the simulation's design parameters. Building on this simulated model, the real-world setup would involve deploying ELiD units at strategic elevated points within an urban road grid, closely replicating the simulated locations for optimal coverage. Each unit would connect to a live backhaul network, linking to edge or cloud servers configured to handle varying data transmission demands through dynamic adjustments to the scan depth parameter δ. The heuristic algorithm would then manage data flow in real time, responding to changing conditions and network congestion to minimize latency. By logging metrics, such as latency, bandwidth, and routing efficiency, across different δ values, this setup would allow for a detailed analysis to refine ELiD placement and routing strategies tailored to dynamic urban environments. Unfortunately, performing the real-world experiments is quite complex due to logistic and financial reasons. Therefore, to test our proposed system, we rely on simulations where we tried to mimic a real-world scenario with realistic parameters.

A. ELiD Placement

We assume our model includes six roads, which are detailed in Table IV. The configuration features the longest main road at the center, surrounded by two parallel roads, and intersected by three perpendicular roads. As given in Table IV, the roads are numbered 1–6, with the main road identified as road 1, the parallel roads as roads 2 and 3, and the perpendicular roads as roads 4 through 6. In addition, itis important to note that the height and rotation of each ELiD are constrained within specific boundaries. We conducted the simulation for two distinct typologies, setting a maximum of 15 ELiDs per road in one scenario, and six ELiDs in the other.

We employ 30 particles and a learning rate of 0.7 to solve the ELiD placement problem using the PSO algorithm. The convergence of the PSO algorithm is demonstrated in Fig. 7(a), where the graph shows that both the fitness and cost functions converge to feasible values while the penalty function declines to zero over successive iterations. This indicates the algorithm's effectiveness in optimizing the placement of ELiDs while adhering to the given constraints. Fig. 7(b) illustrates the ELiD coverage zones on the roadways, with red regions representing areas covered by the ELiD units. The coverage is extensive and strategically placed, ensuring that nearly all the important sectors, as identified in Fig. 7(d), are covered with minimal overlap. This efficient use of ELiD units maximizes coverage and ensures that critical areas are effectively monitored. The final network topology produced by the PSO algorithm is presented in Fig. 7(c). This topology confirms the optimal placement of ELiD units based on sector importance, the maximum number of ELiDs, and the depth parameter. Such strategic placement is crucial for enhancing the overall efficiency and effectiveness of the system. Fig. 7(d) highlights the varying levels of importance for each roadway sector, with higher values signifying more critical areas for coverage. By prioritizing these high-importance sectors, the algorithm ensures that the most crucial areas are adequately monitored, enhancing the system's overall performance. To evaluate the impact of the scan depth parameter on the effectiveness of the coverage area, a sensitivity analysis is shown in Fig. 7(e). The graph illustrates a significant tradeoff between data generation and effective coverage, indicating that higher scan depths lead to more precise data collection for specific locations,

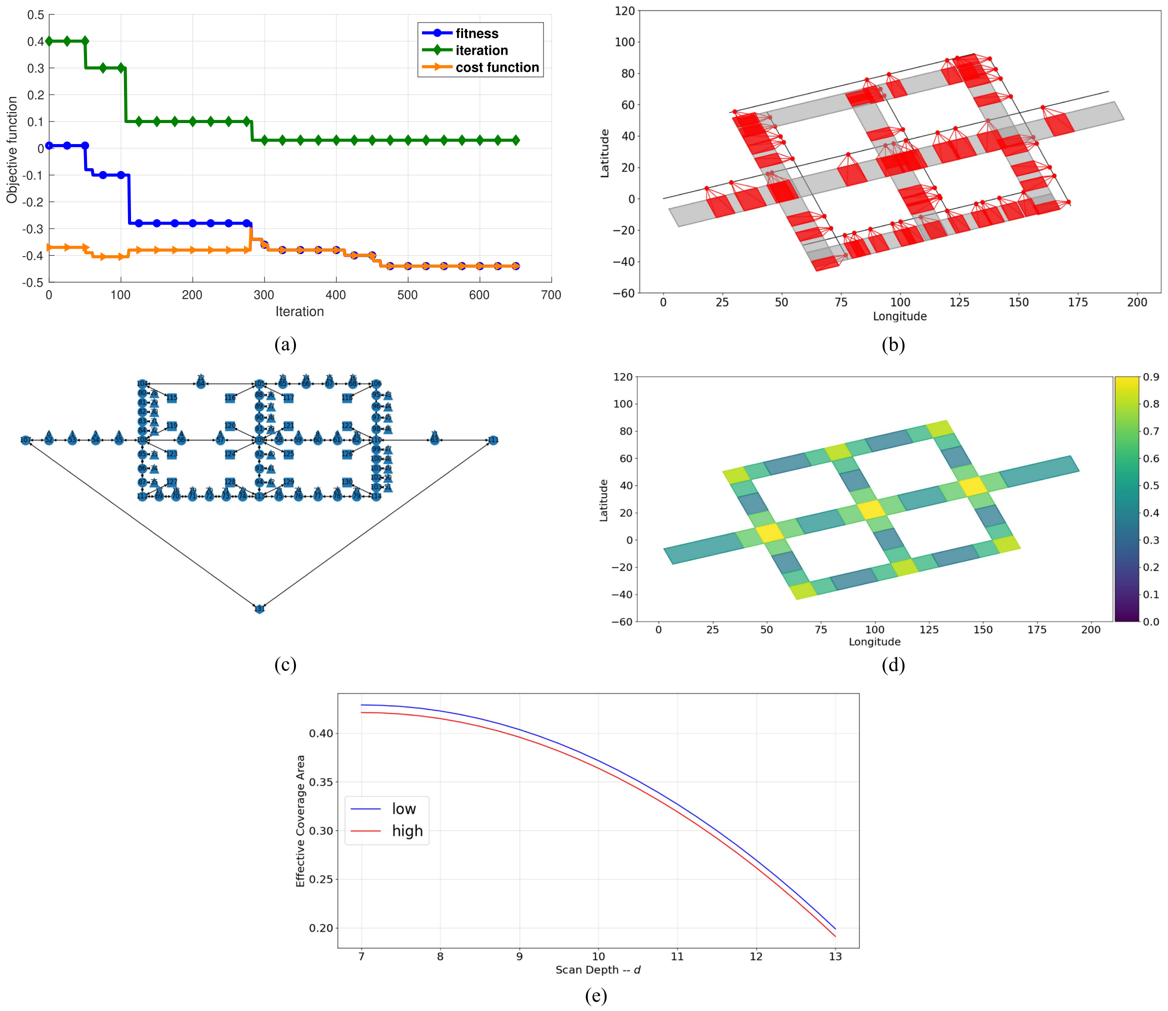


Fig. 7. (a) Convergence of the model's fitness versus the number of iterations. The solid green line represents the penalty function for each iteration. The solid black line corresponds to the model's fitness showing the progress of the objective function, which is the effective coverage area. Finally, the solid orange line represents the cost function for each iteration. (b) ELiD coverage zones on the roadways obtained from the PSO solver solution. (c) Model's topology graph, which is also obtained from the PSO solver results. (d) Importance of each sector for the roadways as an input for the PSO solver. (e) Depth parameter effect on the effective coverage area.

but at the expense of broader spatial coverage. This tradeoff is crucial for balancing data resolution and spatial coverage, as it ensures that the system can provide accurate data where it is most needed while still maintaining a comprehensive overview. By managing this balance, the system can remain both comprehensive and efficient, allowing for optimal data utilization and effective monitoring of critical areas. This balance is essential for applications where both data and wide-area coverage are necessary, highlighting the importance of considering scan depth in the overall system design and operation.

In summary, the convergence graph in Fig. 7(a) demonstrates the PSO algorithm's capability to meet constraints and optimize the objective function effectively. Fig. 7(b) shows that the ELiD coverage zones are extensive and strategically placed, ensuring that critical sectors are well monitored. The network topology in Fig. 7(c) confirms the optimal placement of ELiDs, considering sector importance and constraints. Fig. 7(d) underscores the varying importance of roadway sectors, guiding the placement of ELiDs. Finally, Fig. 7(e) reveals the tradeoff between scan depth and coverage area, crucial for balancing data resolution with spatial coverage.

The algorithm was further tested on a smaller scale by deploying a reduced number of ELiDs and conducting a sensitivity analysis on the scan depth parameter. The results, as presented in Fig. 8, demonstrate the algorithm's adaptability and consistency across varying scales. Fig. 8(a) shows the convergence of the fitness and cost functions over 1200 iterations, indicating that the algorithm efficiently reaches an optimal solution. The penalty function remains constant throughout the iterations, demonstrating that the constraints are consistently met. Fig. 8(b) illustrates the ELiD coverage zones on the roadways, with red regions representing areas covered by the ELiD units. The coverage is thorough and systematically positioned, ensuring effective monitoring of critical areas with minimal overlap, thereby maximizing the utility of each ELiD unit. The final network topology produced by the PSO algorithm is presented in Fig. 8(c). This topology confirms the efficient placement of ELiD units and the connectivity within the network, ensuring that data from

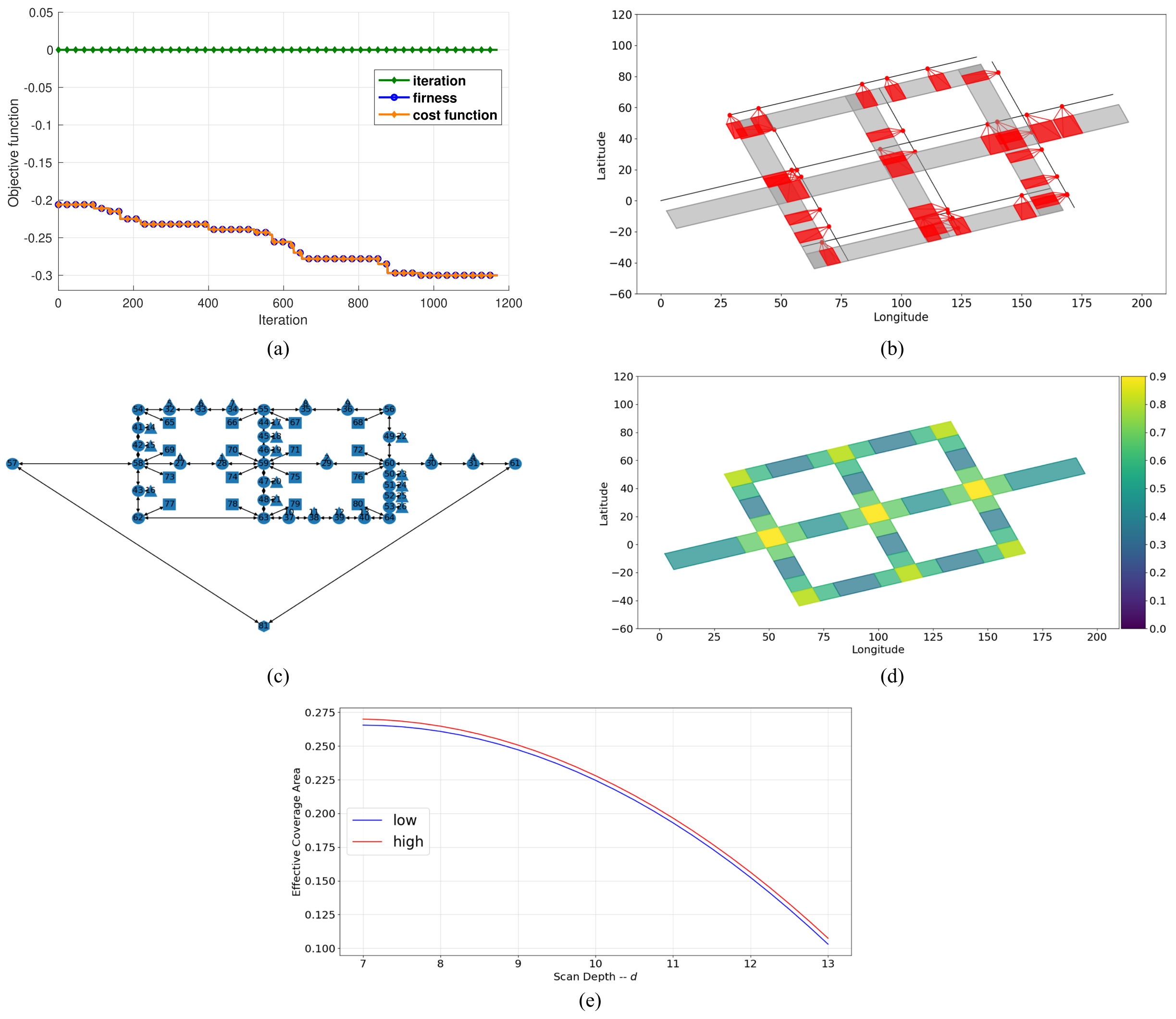


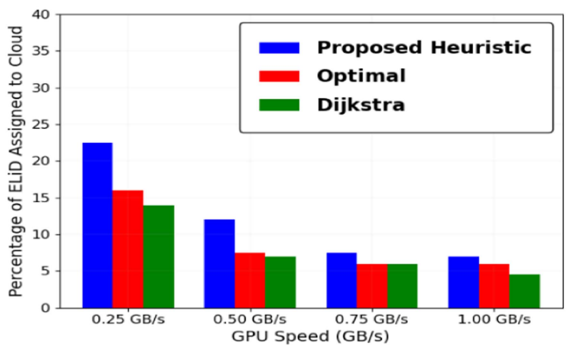
Fig. 8. (a) Convergence of fitness and cost function for the small-scale model. (b) ELiD coverage zones for the smaller model. (c) Solution output topology for the smaller model. (d) Importance of each sector for the smaller model. (e) Depth parameter effect on the smaller model's effective coverage area.

the ELiD units are efficiently routed to processing nodes. This well-structured topology is crucial for optimizing both coverage and data handling capabilities. The sector importance map in Fig. 8(d) indicates varying levels of importance for each roadway sector, with higher values signifying more critical areas for coverage. By prioritizing these high-importance sectors, the algorithm ensures that the most crucial areas are adequately monitored, enhancing the system's overall performance and responsiveness. This targeted approach not only optimizes the deployment of ELiD units but also significantly improves the overall effectiveness of the monitoring system. To evaluate the impact of the scan depth parameter on the effectiveness of the coverage area, a sensitivity analysis is shown in Fig. 8(e). The graph illustrates a tradeoff between data generation and effective coverage, indicating that higher scan depths result in more precise data for specific locations but at the expense of broader spatial coverage. This tradeoff is essential for balancing data resolution and spatial coverage, ensuring that the system remains efficient.

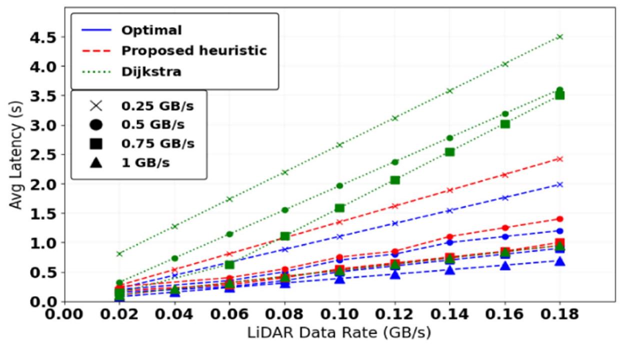
In summary, the results validate the algorithm's robustness across different scales and configurations, making it an adaptable tool for ELiD planning in various urban environments. The convergence behavior, strategic ELiD placement, efficient network topology, prioritized sector coverage, and balanced tradeoffs between data resolution and coverage area demonstrate the algorithm's comprehensive effectiveness. This adaptability ensures that the algorithm can be deployed in diverse urban settings, optimizing LiDAR sensor placement and data processing for enhanced traffic monitoring and AV navigation.

B. Backhaul Routing Optimization

After generating topologies for both the main and smaller models using the PSO algorithm, we applied our heuristic algorithm to the backhaul network to measure latency across various data generation rates and conducted a sensitivity analysis on GPU speed and data generation to examine the impact on ELiD assignment. We compared our heuristic approach with



(a)



(b)

Fig. 9. (a) GPU speed versus ELiD assignment to cloud for the main model. (b) Data rate versus latency for the main model.

TABLE V
GUROBI SOLVER PARAMETERS

Solver 1 parameters	Value	Solver 2 parameters	Value
Heuristics	0.001	Heuristics	default
Gomorypasses	1	Gomorypasses	default
MIRCuts	default	MIRCuts	0
Cutpasses	default	Cutpasses	10
MIPFocus	2	MIPFocus	3
ConcurrentMIP	2	ConcurrentMIP	2

the Gurobi solution regarding convergence time and solution quality. The initial tests were performed on a Windows 11-bit system using Visual Studio and the Windows Subsystem for Linux with Ubuntu for the Linux environment. The computer was equipped with 8 GB of RAM and an 11th Gen Intel Core i5-1135G7 processor. Table V outlines the settings applied in the Gurobi optimizer, where we utilized two concurrent solvers: the first aimed at achieving a feasible solution, and the second focused on sustaining an effective upper bound.

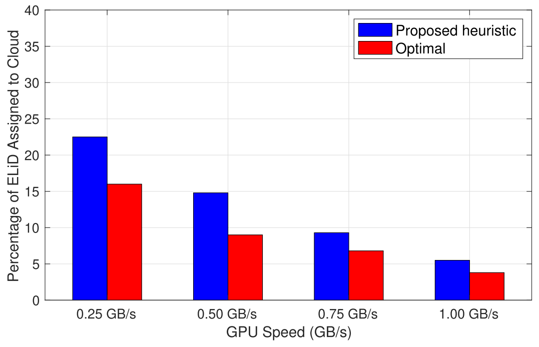
Fig. 9(a) provides a detailed comparison of the performance between the proposed heuristic, the optimal Gurobi solver, and Dijkstra's algorithm. We use Dijkstra's algorithm as a baseline, given its effectiveness in finding the shortest path between nodes in a weighted graph. It calculates the optimal routing paths between LiDAR units and cloud servers within a backhaul network, minimizing data transmission latency. Starting from an initial source node, it progressively selects the lowest-latency routes between nodes, optimizing network performance, reducing delays, and enhancing real-time processing capabilities. In Fig. 9(a), as GPU speed increases, the percentage of ELiDs assigned to the cloud decreases for all solvers. The proposed heuristic consistently assigns a higher percentage at lower GPU speeds, showing its adaptive strategy to utilize cloud resources when local GPU capacity is limited. In contrast, the Gurobi solver adopts a more conservative approach, with fewer ELiDs assigned to the cloud, particularly at lower speeds, although the gap narrows as GPU speed increases.

Fig. 9(b) shows latency trends as LiDAR data rates increase, where Dijkstra consistently performs worse, with the highest latency, indicating inefficiency for high-throughput applications. While Gurobi achieves lower latency at moderate data rates, the gap between Gurobi and the proposed heuristic narrows at higher data rates, highlighting the heuristic's competitive edge for real-time applications. Average latency metrics reveal distinct characteristics, with the Gurobi solver consistently achieving the lowest latency (0.22–5.19 s), followed closely by the heuristic (0.81–7.33 s). Despite slightly higher latency, the heuristic offers flexibility and adaptability in resource management, making it a compelling alternative for time-sensitive, dynamic environments. Dijkstra's highest latencies (up to 4.5 s) make it less suitable for such applications.

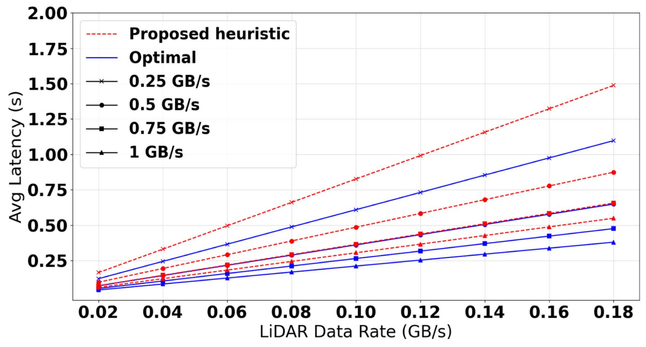
In Fig. 10(a), the percentage of ELiDs assigned to the cloud as GPU speed increases is analyzed. At lower GPU speeds (0.25 GB/s), the proposed heuristic assigns more ELiDs to the cloud (16%) compared to the optimal solver (22.5%). As GPU speed increases, both methods show a similar decline in cloud assignments, reaching about 6%–7% at 1.00 GB/s. The heuristic solution leverages cloud resources more at lower speeds, while the optimal solver adopts a more measured approach. The gap between the two methods reduces as GPU speed increases, suggesting that both algorithms become more efficient at utilizing local processing power.

Fig. 10(b) illustrates the impact on latency as LiDAR data rates increase. Here, the proposed heuristic again demonstrates slightly higher latencies compared to the optimal solver across all GPU speeds. The latency differences are minimal at lower data rates but become more pronounced at higher data rates (close to 0.18 GB/s). Both methods show a linear increase in latency as the data rate increases, with the proposed heuristic maintaining a competitive latency profile. The heuristic's performance remains close to the optimal solution, especially at higher GPU speeds, demonstrating its effectiveness for real-time applications, despite the slight latency overhead.

The analysis of both the main and smaller models highlights the strength of the proposed heuristic in balancing efficiency, adaptability, and response times. It assigns more ELiDs to the cloud at lower GPU speeds while remaining competitive in terms of latency. Although the optimal solver achieves the lowest latency, the heuristic closes the gap at higher data rates, making it valuable for real-time applications. Dijkstra's high latency makes it unsuitable for high-throughput scenarios. Overall, the proposed heuristic is an effective compromise, ideal for dynamic



(a)



(b)

Fig. 10. (a) GPU speed versus ELiD assignment to cloud for the smaller model. (b) Data rate versus latency for the smaller model.

TABLE VI
CONVERGENCE TIME OF EACH SOLVER FOR THE MAIN MODEL IN SECONDS (S)

Solver	Min	Max	Mean
Heuristic	93.314	133.123	101.898
Optimal	119.001	395.098	243.241
Dijkstra	91.180	523.000	307.09

TABLE VII
CONVERGENCE TIME OF EACH SOLVER FOR THE SMALLER MODEL IN SECONDS (S)

Solver	Min	Max	Mean
Heuristic	26.097	93.885	33.783
Optimal	48.411	207.749	106.456

environments requiring fast decision-making and adaptable resource allocation.

The convergence times for the three solvers reveal significant differences in their computational complexity, which is presented in Table VI. The heuristic method demonstrates the fastest convergence, with a mean time of approximately 101.898 s, reflecting its efficiency in finding solutions quickly. This performance is particularly advantageous in scenarios requiring rapid decision-making. In contrast, the optimal Gurobi solver exhibits a notably higher mean convergence time of 243.241 s, with maximum times reaching 395 s. While this solver guarantees optimal solutions, its increased computational complexity results in longer convergence periods, which can be a drawback in real-time applications. The Dijkstra algorithm shows the highest variability, with a maximum convergence time of 523 s and a mean of 307 s. This indicates a significant computational burden, making it less suitable for applications that demand timely responses. Overall, the heuristic approach not only converges the fastest but also showcases lower complexity compared to both the optimal and Dijkstra algorithms, underscoring its practicality for applications that prioritize speed over absolute optimality.

Table VII presents the convergence time statistics for heuristic and optimal solvers in the smaller model. The heuristic solver, with a minimum convergence time of 26.097, a maximum of 93.885, and a mean of 33.783, demonstrates quicker convergence overall compared to the optimal solver. The optimal solver, while offering a higher degree of precision, shows a significantly higher minimum (48.411), maximum (207.749), and mean (106.456) convergence time, reflecting the tradeoff between solution accuracy and computational efficiency. This suggests that while the heuristic solver may be less precise, it provides faster results.

C. Real-World Implementation and Practical Challenges

Our simulation results validate the efficiency of using an optimized ELiD-MEC infrastructure to support traffic monitoring and AV navigation. However, validating these results via real-world implementation would be very beneficial for practitioners. However, this is a very elaborate task, which is left for a future extension of this work. In this section, we discuss and present some recommendations for practical implementation of the system and highlight related challenges. Hence, it is recommended to plan the deployment of ELiD units in an optimized manner as presented in Section IV-A in the area of interest (e.g., road segments or intersections) while taking into account key factors, such as traffic patterns, congestion level, and road dimensions. Live data collection could then be performed where ELiD sensors capture real-time LiDAR data (e.g., collecting point cloud of the monitored areas), which will then be offloaded to the installed MEC servers for immediate processing. Different wired and wireless paths (i.e., potential routes) could be established between the ELiDs and the MEC servers according to the budget availability. Then, the offload strategy presented in Section V-B could be performed to route the data over the network, which could integrate fiber-optic connections or wireless transmission. While the system is operating, its ability to handle dynamic urban challenges, such as fluctuating network loads, voluminous data, and the variability of pedestrian and vehicular traffic, will be assessed. The performance will be monitored in terms of latency, data accuracy, processing speed, and the reliability of processed outputs that could be shared with a test vehicle.

Deploying the ELiD MEC system in the real world would face challenges in infrastructure setup, data management, and environmental reliability. EliD with MECs would require stable power, robust network connectivity, and weather resistance. Managing high data volumes while maintaining low latency for real-time processing is critical, and algorithms may require tuning to handle real-world data variability. Security and privacy concerns, especially in urban areas, would demand strong safeguards, while operational costs and maintenance needs could impact long-term feasibility. Together, these challenges make real-world implementation difficult due to significant financial and logistical demands.

Nevertheless, a real-world validation will provide comprehensive insights into the ELiD-MEC framework's scalability, reliability, and effectiveness, enabling adjustments and optimizations that are crucial for its implementation in smart cities. These steps will ensure that the system not only meets the performance requirements to support collaborative autonomous navigation, but also proves its viability for large-scale deployment in urban areas.

VII. Conclusion

In conclusion, our study presents a comprehensive examination of the integration of ELiD with MEC for traffic monitoring in smart city infrastructures. The findings reveal that our proposed heuristic approach successfully reduces reliance on cloud processing, promoting computational efficiency at the network's edge. This approach marginally increases latency but maintains a small optimality gap compared to the best possible solution, which suggests that the heuristic is an effective strategy for real-time LiDAR data processing in urban environments. The research highlights the delicate balance between latency, cloud resource utilization, and computational expediency, providing a viable blackprint for future ITS. While our current results are based on simulations, the next important step is to validate the proposed ELiD-MEC framework through real-world experiments at least for a small version of the system. Furthermore, we aim to exploit the framework to leverage collaborative driving techniques, such as collaborative perception and traffic flow estimation.

References

[1] M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang, "Wireless sensor networks in intelligent transportation systems," *Wireless Commun. Mobile Comput.*, vol. 9, no. 3, pp. 287–302, 2009.

[2] O. Rinchi, I. Shatnawi, and A. Alsharoa, "Deep-learning-based accurate beamforming prediction using LiDAR-assisted network," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, Toronto, Canada, 2023, pp. 1–5.

[3] T. Luettel, M. Himmelsbach, and H. Wuensche, "Autonomous ground vehicles—concepts and a path to the future," *Proc. IEEE*, vol. 100, pp. 1831–1839, May 2012.

[4] M. Senapati, B. Anand, A. Thakur, H. Verma, and P. Rajalakshmi, "Object detection and segmentation using LiDAR-camera fusion for autonomous vehicle," in *Proc. 5th IEEE Int. Conf. Robot. Comput.*, Taichung, Taiwan, 2021, pp. 123–124.

[5] B. Cherif, H. Ghazzai, A. Alsharoa, H. Besbes, and Y. Massoud, "Aerial LiDAR-based 3D object detection and tracking for traffic monitoring," in *Proc. IEEE Int. Symp. Circuits Syst.*, Monterey, CA, USA, 2023, pp. 1–5.

[6] Z. Osterwisch, O. Rinchi, A. Alsharoa, H. Ghazzai, and Y. Massoud, "Multiple UAV-LiDAR placement optimization under road priority and resolution requirements," in *Proc. IEEE Int. Conf. Commun.*, Rome, Italy, 2023, pp. 241–246.

[7] X. Zhang, H. Zhang, C. Qian, B. Li, and Y. Cao, "A LiDAR-intensity SLAM and loop closure detection method using an intensity cylindrical-projection shape context descriptor," *Int. J. Appl. Earth Observ. Geoinformation*, vol. 122, 2023, Art. no. 103419.

[8] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, "A comparative analysis of LiDAR SLAM-based indoor navigation for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6907–6921, Jul. 2022.

[9] M. C. Lucic, H. Ghazzai, A. Alsharoa, and Y. Massoud, "A latency-aware task offloading in mobile edge computing network for distributed elevated LiDAR," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sevilla, Spain, 2020, pp. 1–5.

[10] Driverless cars are giving engineers a fuel economy headache. Bloomberg.com, 2017. Accessed: Oct. 10, 2019. [Online]. Available: https://www.bloomberg.com/news/articles/2017-10-11/driverless-cars-are-givingengineers-a-fuel-economy-headache.

[11] Low complexity real-time simultaneous localization and mapping using velodyne LiDAR sensor, Mar. 2018. Accessed: Oct. 10, 2019. [Online]. Available: http://ondemand.gputechconf.com/gtc/2018/presentation/s8501-low-complexity-real-timesimultaneous-localization-and-mapping-using-velodyne-lidar-sensor.pdf

[12] J. Huang, J. Stoter, R. Peters, and L. Nan, "City3D: Large-scale building reconstruction from airborne LiDAR point clouds," *Remote Sens.*, vol. 14, no. 9, 2022, Art. no. 2254.

[13] Y. Liu, A. Obukhov, J. D. Wegner, and K. Schindler, "Point2building: Reconstructing buildings from airborne LiDAR point clouds," *ISPRS J. Photogrammetry Remote Sens.*, vol. 215, pp. 351–368, 2024.

[14] Y. Li and J. Ibanez-Guzman, "LIDAR for autonomous driving: The principles, challenges, and trends for automotive LIDAR and perception systems," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 50–61, Jul. 2020.

[15] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," presented at the 29th USENIX Security Symposium, 2020. Accessed: Oct. 10, 2019. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/sun

[16] A. Poulose, M. Baek, and D. S. Han, "Point cloud map generation and localization for autonomous vehicles using 3D LIDAR scans," in *Proc. 27th Asia Pacific Conf. Commun.*, Jeju Island, Korea, 2022, pp. 336–341.

[17] H. Hu, Z. Liu, S. Chitlangia, A. Agnihotri, and D. Zhao, "Investigating the impact of multi-LiDAR placement on object detection for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, New Orleans, Louisiana, USA, 2022, pp. 2540–2549.

[18] V. Pritzl, M. Vrba, P. Štěpán, and M. Saska, "Cooperative navigation and guidance of a micro-scale aerial vehicle by an accompanying UAV using 3D LiDAR relative localization," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, Dubrovnik, Croatia, 2022, pp. 526–535.

[19] N. Jayaweera, N. Rajatheva, and M. Latva-aho, "Autonomous driving without a burden: View from outside with elevated LiDAR," in *Proc. IEEE 89th Veh. Technol. Conf.*, Kuala Lumpur, Malaysia, 2019, pp. 1–7.

[20] M. C. Lucic, H. Ghazzai, and Y. Massoud, "A generalized dynamic planning framework for green UAV-assisted intelligent transportation system infrastructure," *IEEE Syst. J.*, vol. 14, no. 4, pp. 4786–4797, Dec. 2020.

[21] "What you need to know about the world's water wars," *Nat. Geographic*, 2016. Accessed: Oct. 10, 2019. [Online]. Available: https://www.nationalgeographic.com/news/2016/07/world-aquifers-water-wars/

[22] D. Deibe, M. Amor, and R. Doallo, "Big data storage technologies: A case study for web-based LiDAR visualization," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 3831–3840.

[23] D. Ardagna, "Cloud and multi-cloud computing: Current challenges and future applications," in *Proc. IEEE/ACM 7th Int. Workshop Princ. Eng. Serv.-Oriented Cloud Syst.*, Florence, Italy, 2015, pp. 1–2.

[24] A. Bader, H. Ghazzai, A. Kadri, and M. Alouini, "Front-end intelligence for large-scale application-oriented Internet-of-Things," *IEEE Access*, vol. 4, pp. 3257–3272, 2016.

[25] X. Wei et al., "MVR: An architecture for computation offloading in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput.*, Honolulu, Hawaii, USA, 2017, pp. 232–235.

[26] Q. Zhang, H. Zhong, J. Wu, and W. Shi, "How edge computing and initial congestion window affect latency of web-based services: Early experiences with Baidu?," in *Proc. IEEE/ACM Symp. Edge Comput.*, Bellevue, WA, USA, 2018, pp. 393–398.

[27] S. Thornton and S. Dey, "Multi-modal data and model reduction for enabling edge fusion in connected vehicle environments," *IEEE Trans. Veh. Technol.*, vol. 73, no. 8, pp. 11979–11994, Aug. 2024.

[28] D. Loghin, L. Ramapantulu, and Y. M. Teo, "Towards analyzing the performance of hybrid edge-cloud processing," in *Proc. IEEE Int. Conf. Edge Comput.*, Milan, Italy, 2019, pp. 87–94.

[29] P. Kong, "Computation and sensor offloading for cloud-based infrastructure-assisted autonomous vehicles," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3360–3370, Sep. 2020.

[30] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based Internet of Vehicle infrastructure for autonomous driving," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 149–160, Feb. 2019.

[31] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *Proc. 55th Annu. Conf. Soc. Instrum. Control Engineers Jpn.*, Tsukuba, Japan, 2016, pp. 1122–1127.

[32] R. Santos, H. Ghazzai, and A. Kassler, "Optimal steerable mmWave mesh backhaul reconfiguration," in *Proc. IEEE Glob. Commun. Conf.*, Abu Dhabi, UAE, 2018, pp. 1–7.

[33] A. Mathew, T. Das, P. Gokhale, and A. Gumaste, "Multi-layer high-speed network design in mobile backhaul using robust optimization," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 4, pp. 352–367, Apr. 2015.

[34] G. Nie, H. Tian, and J. Ren, "Energy efficient forward and backhaul link optimization in OFDMA small cell networks," *IEEE Commun. Lett.*, vol. 19, no. 11, pp. 1989–1992, Nov. 2015.

[35] H. Ghazzai, H. Menouar, and A. Kadri, "On the placement of UAV docking stations for future intelligent transportation systems," in *Proc. IEEE 85th Veh. Technol. Conf.*, Sydney, Australia, 2017, pp. 1–6.

[36] M. C. Lucic, H. Ghazzai, and Y. Massoud, "Elevated LiDAR placement under energy and throughput capacity constraints," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst.*, 2020, pp. 897–900.

[37] The architectural implications of autonomous driving: Constraints and acceleration, 2018. Accessed: Oct. 10, 2019. [Online]. Available: https://web.eecs.umich.edu/~shihclin/papers/AutonomousCar-ASPLOS18.pdf

[38] S. Kumar, S. Gollakota, and D. Katabi, "A cloud-assisted design for autonomous driving," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, Helsinki, Finland, 2012, pp. 41–46.

[39] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2131–2165, Fourth Quarter 2021.

[40] L. A. Haibeh, M. C. E. Yagoub, and A. Jarray, "A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches," *IEEE Access*, vol. 10, pp. 27591–27610, 2022.

[41] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization*, vol. 108. Philadelphia, PA, USA: SIAM, 2009.

[42] A. D. Pia, S. S. Dey, and M. Molinaro, "Mixed-integer quadratic programming is in NP," *Math. Program.*, vol. 162, no. 1, pp. 225–240, 2017.

[43] W. A. Chaovalitwongse, P. M. Androulakis, and P. I. Pardalos, "Quadratic integer programming: Complexity and equivalent forms," in *Encyclopedia of Optimization*, New York, NY, USA: Springer, 2020, pp. 1–8.

[44] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Perth, WA, Australia, 1995, pp. 1942–1948.

[45] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. Comput. Cybern. Simul.*, Orlando, FL, USA, 1997, pp. 4104–4108.

[46] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, 1959.

[47] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.