Data-Driven Capacity Planning for Vehicular Fog Computing

Wencan Mao, Ozgur Umut Akgul, Abbas Mehrabi, *Member, IEEE*, Byungjin Cho, Yu Xiao, *Member, IEEE*, and Antti Ylä-Jääski

***Abstract*—The strict latency constraints of emerging vehicular applications make it unfeasible to forward sensing data from vehicles to the cloud for processing. To shorten network latency, vehicular fog computing (VFC) moves computation to the edge of the Internet, with the extension to support the mobility of distributed computing entities [a.k.a fog nodes (FNs)]. In other words, VFC proposes to complement stationary FNs co-located with cellular base stations with mobile ones carried by moving vehicles (e.g., buses). Previous works on VFC mainly focus on optimizing the assignments of computing tasks among available FNs. However, capacity planning, which decides where and how much computing resources to deploy, remains an open and challenging issue. The complexity of this problem results from the spatiotemporal dynamics of vehicular traffic, varying computing resource demand generated by vehicular applications, and the mobility of FNs. To solve the above challenges, we propose a data-driven capacity planning framework that optimizes the deployment of stationary and mobile FNs to minimize the installation and operational costs under the quality-of-service constraints, taking into account the spatiotemporal variation in both demand and supply. Using real-world traffic data and application profiles, we analyze the cost efficiency potential of VFC in the long term. We also evaluate the impacts of traffic patterns on the capacity plans and the potential cost savings. We find that high traffic density and significant hourly variation would lead to dense deployment of mobile FNs and create more savings in operational costs in the long term.**

***Index Terms*—Application profiling, capacity planning, integer linear programming (ILP), spatiotemporal analysis, technoeconomic analysis, vehicular fog computing (VFC).**

I. Introduction

CLOUD computing has long been the dominant solution for handling the big data generated from various sources [1]. However, the traditional cloud strategies are not feasible for the emerging vehicular applications with strict latency constraints, such as cooperative intersection crossing [2] and lane change scheduling [3] for autonomous vehicles. Fog computing, as a promising alternative, moves computation resources to the edge of the network [4] and reduces network latency due to its close proximity to the end users and dense geographical distribution [5].

In the fog computing scenarios, distributed fog computing entities, often called fog nodes (FNs), can be installed in network infrastructures, such as cellular base stations (BSs) and road side units (RSUs). We call these cellular FNs (CFNs). This stationary deployment of FNs often forces service providers to overprovision the resources to ensure the Quality-of-Service (QoS) requirements and turn the service provisioning into a nonprofitable business model. Motivated by this technoeconomic pressure, vehicular fog computing (VFC) has been proposed to complement stationary FNs with mobile ones, which are carried by vehicles, such as buses, taxis, and drones. We call them vehicular FNs (VFNs). VFNs provide computing services to the surrounding vehicles within the range of single-hop V2X communications. With the mobility of VFNs, it becomes possible to satisfy the dynamic resource demand in a more cost-efficient manner [6].

Previous works on VFC have mainly focused on the task assignment problem among available FNs using various methods, including reinforcement learning and particle swarm optimization. For example, a joint optimization solution was developed in [7] to assign the tasks generated from vehicles across the stationary and mobile FNs under the constraints of service latency, quality loss, and fog capacity. Unlike the task allocation problem, capacity planning focuses on determining the locations and capacities of FNs. A capacity planning framework was proposed in [8] to satisfy the QoS requirements while minimizing the number of required FNs. However, this framework considers only stationary deployment of FNs and therefore, cannot be applied to VFC.

Despite the increasing focus on the VFC capacity planning problem, it is still a challenging issue. First, vehicular traffic has high spatiotemporal diversity, where the traffic flow depends on the time of day and the geographic location [9]. The capacity planning for VFC requires a deep understanding of the spatiotemporal dynamics of vehicular traffic. Second, to estimate the computing resource demand, it is necessary to consider the resource consumption pattern of various vehicular applications. Finally, VFNs are supposed to serve the vehicles within the one-hop communication range. The mobility of VFNs adds a layer of

Manuscript received 6 May 2021; revised 23 September 2021 and 22 November 2021; accepted 3 January 2022. Date of publication 18 January 2022; date of current version 25 July 2022. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 825496 and Grant 815191, and in part by the Academy of Finland under Grant 317432 and Grant 318937. *(Corresponding author: Wencan Mao.)*

Wencan Mao is with the Department of Communications and Networking, Aalto University, 02150 Espoo, Finland (e-mail: wencan.mao@aalto.fi).

Ozgur Umut Akgul, Byungjin Cho, and Yu Xiao are with the Department of Communications and Networking, Aalto University, 02150 Espoo, Finland.

Abbas Mehrabi is with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K.

Antti Ylä-Jääski is with the Department of Computer Science, Aalto University, 02150 Espoo, Finland.

Digital Object Identifier 10.1109/JIOT.2022.3143872

complexity to the analysis of service availability and cost estimation.

In this article, we propose a data-driven capacity planning framework that takes real-world traffic data and application profiles as inputs and outputs a cost-optimal deployment plan of CFNs and VFNs using a heuristic algorithm and integer linear programming (ILP). Our framework determines the number and types of FNs needed in different regions to satisfy the computing resource demand and plans the trajectory and timetables of VFNs based on real-world bus schedules. The contributions of this work are as follows.

1) To the best of our knowledge, this is the first work on data-driven capacity planning for VFC, taking into account the spatiotemporal distribution of computing resource demand and supply. It provides a mathematical model for minimizing the installation and operational costs under QoS constraints of emerging vehicular applications.
2) Through evaluation with real-world data sets, we show the potential of complementing CFNs with VFNs to fulfill the dynamic computing resource demand with lower costs. Our evaluation also indicates that the deployment of VFNs would increase the service rate by up to 9.3% compared to the scenario where only CFNs are used.
3) Our study provides deep insights on the impacts of traffic patterns on the FN deployment strategies: we compare the deployment plan and the potential cost savings between areas with different traffic characteristics (e.g., a downtown area with dense traffic flows versus a suburb area with low traffic volume), and between weekdays and weekends. We discuss the uncertainty in the demand and the ways to handle it.

The remainder of this article is organized as follows. Section II presents the related works. Section III overviews VFC. Section IV introduces the data-driven methodology of the capacity planning platform for VFC. Section V presents methods for estimating computing resource demand based on real-world traffic data and application profiles. Section VI formulates the optimization problem for capacity planning. After introducing the experimental setup in Section VII, the functionality evaluation and results analysis are discussed in Sections VIII and IX, respectively. Finally, Section X presents the discussion before we conclude the work in Section XI.

II. Related Work

In this section, we compare our work with the state-of-the-art methods in spatiotemporal analysis, task allocation, and capacity planning.

A. Spatiotemporal Analysis

Previous research on spatiotemporal analysis mainly focused on finding the spatiotemporal patterns (i.e., traffic status, interaction among road segments, and changing trends) of the vehicular traffic. For example, Zhang *et al.* [10] employed the dictionary-based compression theory to detect the anomaly behavior in road networks by analyzing the multidimensional traffic data. Zhang *et al.* [11] proposed a multiagent system to examine the spatiotemporal characteristics of the traffic data and the cooperation and workflow among the road segments. Our work not only analyzes the spatiotemporal patterns but also uses the regression-based methods and traffic flow theory to model the vehicular traffic.

B. Task Allocation

Previous works used both classic methods (e.g., greedy algorithm) and new methods (e.g., reinforcement learning) for task allocation in the edge/fog computing environment. Sahni *et al.* [12] proposed a data-aware multistage greedy adjustment algorithm to schedule tasks and network flows together to achieve low latency. Gu *et al.* [13] designed a distributed and context-aware task assignment mechanism to reduce overall energy consumption while satisfying the heterogeneous delay requirements. Wang *et al.* [14] presented a latency-aware heterogeneous mobile-edge computing system, where the data are offloaded to the cloud center if the edge cannot process it on time. Mai *et al.* [15] developed a reinforcement learning approach that utilizes the evolution strategies for real-time task assignment among FNs to minimize the total latency during a long-term period. These works have focused on assigning the tasks to the stationary FNs without considering the mobility of vehicles.

Recently, new methods, such as particle swarm optimization and reinforcement learning, have been used for task allocation in VFC, taking the mobility of vehicles into account. Zhu *et al.* [7] designed a dynamic task allocation framework where binary particle swarm optimization was used to jointly optimize the quality and latency. Hou *et al.* [23] proposed a software-defined networking and edge computing-aided Internet of vehicle framework, where they use fault-tolerant particle swarm optimization for the partial computation offloading and reliable task allocation. Zhou *et al.* [24] presented a two-stage VFC framework with a contract theory-based vehicular computational resource management mechanism and a matching-learning-based task offloading mechanism. Zhu *et al.* [25] proposed a latency and resolution-aware task offloading framework based on a partially observable Markov decision process. They further proposed a deep Q-network to learn the optimized task allocation strategies for increasing the Quality of Information (QoI) of collected data while reducing the processing latency [26]. Apart from above, a deep reinforcement learning-based algorithm was designed in [27] for maximizing both the expected reward and the entropy of policy, while simultaneously evaluating the service availability of VFNs. A vehicular sensing networks-aided smart city model was proposed in [28] with an information source selection algorithm and a reinforcement learning-based city information sharing mechanism.

Although task allocation and capacity planning are usually coupled together for managing the computing resources in VFC, they focus on different targets. The methods above focused on how to assign the tasks to the available FNs, whereas our work focuses on where to deploy the FNs and how much computing capacity is required to meet the demand.

TABLE I
COMPARISON OF OUR WORK WITH OTHER CAPACITY PLANNING METHODS

Work	Method	Objective	Outputs	Constraints	User Mobility	Resource Mobility
[16]	Heuristic algorithm	Minimize latency	A feasible set of FNs	Communication, processing data, computing tasks	No	No
[17]	Weighted sum method, evolutionary algorithms	Minimize latency and costs	Location, number, capacity of FNs	Source, openness, fog type, link type, fog capacity, link capacity	No	No
[18]	Weighted sum method, hierarchical, trade-off	Minimize latency and cloud traffic	Location, number, capacity of FNs	Node capacity, network capacity, cloud network capacity, costs	No	No
[19]	Mixed integer linear programming	Minimize costs	Routing strategy among BSs and data centers	Probabilistic delay guarantees	No	No
[8]	Knapsack algorithm	Improve edge utilization	Combinations of resource demands	Latency, resource consumption	No	No
[20]	Queuing model	Minimize the number of FNs	Number of FNs	Latency, resource consumption	No	No
[21]	Integer linear programming	Minimize latency and energy	Location, number, capacity of FNs	Node capacity, network capacity, link type, cloud network capacity	Yes	No
[22]	Mixed integer linear programming	Minimize costs	Location, number, power of RSUs	Network coverage, computational demand	Yes	No
Ours	Heuristic, integer linear programming	Minimize costs	Location, number of FNs, and schedules of VFNs	Latency, resource consumption, capacity, availability	Yes	Yes

C. Capacity Planning

The capacity planning in the edge/fog computing environment is generally formulated as an optimization problem with different objectives, inputs, outputs, and constraints. Table I presents a comparison between our work and the state-of-the-art capacity planning methods. Due to space limitations, we have not detailed the inputs, but they can be referred to in the works listed in the table.

One way to model and solve the capacity planning problem is to use classical methods (e.g., Knapsack algorithm). For example, Noreikis *et al.* [8] established a knapsack-based capacity planning model for edge computing aiming to satisfy the QoS requirements while minimizing the number of required FNs. They further employed the queuing theory for the capacity planning for the real-time computing-intensive applications [20]. However, these methods cannot be applied directly to VFC, because they did not consider the mobility of the vehicles, including the ones generating computing demand and the ones carrying the computing resources.

Alternatively, we can use heuristics for capacity planning. For example, Chiu *et al.* [16] utilized a heuristic algorithm to simultaneously decide the number of FNs with proper communication resource allocation and computing task assignment. Zhang *et al.* [17] presented a framework that employed weighted sum and evolutionary algorithms to optimize the tradeoff between the capital expenditure and the network delay. Haider *et al.* [18] used weighted sum, hierarchical, and trade-off methods to simultaneously determine the optimal location, capacity, and the number of FNs, as well as the connection between the FNs and the cloud to minimize the delay in the network and the traffic to the cloud. Despite providing a fast result, the heuristic solutions are usually suboptimal.

Another solution for capacity planning is to use ILP or mixed-ILP (MILP). For example, Stypsanelli *et al.* [19] proposed an optimal capacity planning solution for fog computing infrastructures under probabilistic delay guarantees aiming to save energy and operations costs. Hussain *et al.* [21] aimed to find the optimal location and capacity of FNs toward minimizing overall network delay and energy consumption. Premsankar *et al.* [22] aimed to minimize the deployment cost of edge devices by jointly satisfying a target level of network coverage and computational demand of vehicular applications in smart cities. By using ILP or MILP, we can guarantee the optimal solution. However, the execution time can be lengthy, especially when the computational complexity is high.

As summarized in Table I, we utilize both a heuristic algorithm and ILP for the capacity planning for VFC. We use a heuristic to rapidly estimate the upper bound of the computing demand generated by the vehicular applications, and we use ILP to obtain the optimal deployment decisions of CFNs and VFNs. Moreover, our solution differs from previous works by supporting the mobility of FNs and utilizing it to maximize cost efficiency. The outputs of our algorithm include not only the locations and capacities of CFNs but also the schedules and capacities of VFNs.

III. VEHICULAR FOG COMPUTING

In this section, we present an overview of the VFC paradigm. First, we demonstrate an application scenario of VFC. Then, we introduce the vehicular communication technologies to support the implementation of VFC.

A. VFC Application Scenario

Fig. 1 presents an application scenario of VFC. In this scenario, Vehicle A generates an object detection task to recognize the traffic signs. It is within the communication range of a bus that carries a VFN. Thus, Vehicle A offloads the task to the bus. Meanwhile, Vehicle B generates a lane detection task, which is offloaded to a CFN co-located with the connected cellular BS.

In case more than one FN is available within the vehicle's communication range, the vehicle uses task allocation algorithms to decide where to offload the tasks. Capacity planning, on the other hand, focuses on planning where to deploy the FNs and how much computing capacity should be

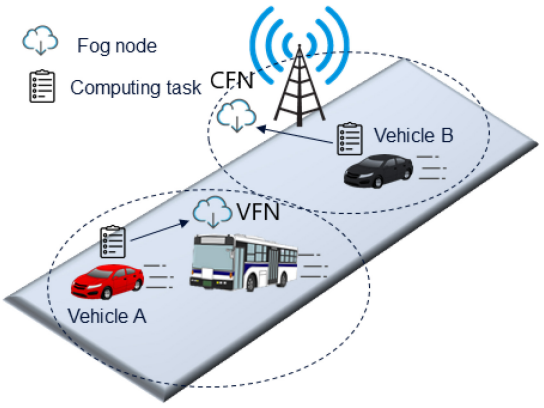

Fig. 1. Application scenario of VFC, where Vehicle A offloads its task to a VFN carried by a bus and Vehicle B offloads its task to a CFN co-located with a cellular BS.

deployed to fulfill the estimated computing resource demand with better technoeconomic performance. In this article, we focus on capacity planning for VFC. We employ CFNs for fulfilling the constant demand while utilizing VFNs to meet the spatial–temporal varying demand from the vehicular traffic environment.

B. Vehicular Communication Technology

Dedicated short-range communications (DSRCs) and cellular V2X (C-V2X) are the most widely used radio access technologies for vehicular communication. DSRC uses an orthogonal frequency-division multiplexing-based physical layer with a channel bandwidth of 10 MHz [29]. C-V2X uses the widely distributed cellular infrastructure and defines additional transmission modes that allow direct V2X communication using side-link channels [29]. According to [30], DSRC and C-V2X can support basic safety applications (e.g., advertising driving alerts periodically) as long as the vehicular density is not intense. The latency requirements for these applications are 100 ms [29].

Moreover, IEEE 802.11bd and 5G NR V2X are designed to support the vehicular applications characterized by high-reliability and low-latency requirements [29]. These applications, so-called advanced vehicular applications, aim to increase driving safety and benefit traffic management. 3GPP has divided these applications into four categories: 1) vehicle platooning; 2) advanced driving; 3) extended sensor; and 4) remote driving. Their latency requirements are 10–500, 3–100, 3–100, and 5 ms, respectively [31].

In this work, we consider 5G NR V2X as the communication module among the vehicles, which enables vehicular communications either within or out of the gNodeB coverage and supports multiple communication types (i.e., broadcast, groupcast, and unicast) and message types (i.e., periodic and aperiodic). A system-level evaluation of the 5G NR V2X is provided by [32] using a 60-kHz subcarrier spacing, a 20-MHz channel, and a 10-Hz transmission rate. Under highway scenarios, the packet delivery rate (PDR) is between 99.7% and 99.8%; under urban scenarios, the PDR varies from 93% to 97% [32]. Consequently, 5G NR V2X can provide high-reliable vehicular communication for VFC.

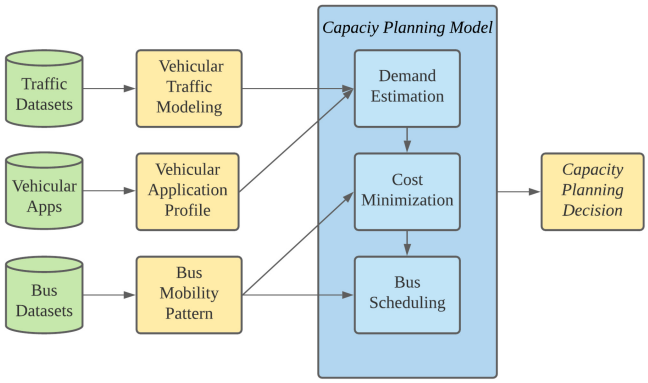

Fig. 2. Flowchart of data-driven capacity planning.

IV. Data-Driven Capacity Planning

We follow a data-driven methodology to plan for the deployment solutions of CFNs and VFNs. Fig. 2 presents an overview of the data-driven capacity planning process. We implement the capacity planning model in three steps: 1) demand estimation; 2) cost minimization; and 3) bus scheduling. In this model, we use three types of data as the inputs. The vehicular traffic data and application profiles are used for demand estimation, and the bus mobility data are used for cost minimization and bus scheduling.

We first estimate the computing resource demand generated by the vehicles, which depends on the spatiotemporal distribution of vehicular traffic and the resource consumption profiles of vehicular applications. The latter describes the usage pattern of central processing unit (CPU) and graphics processing unit (GPU) resources for each application. Based on real-world traffic data sets, we apply spatiotemporal analysis methods, such as clustering, traffic flow theory, and Gaussian process regression, to model traffic flows. Meanwhile, we choose a set of representative vehicular applications as examples and profile their resource usages under different latency constraints. The output of the demand estimation module defines the minimum amount of computing capacity (in terms of the number of FNs with fixed unit size) required in each cluster to meet the QoS requirements.

Our second step is to determine a cost-optimal deployment plan based on the estimated demand and the potential supply. We assume that VFNs would be installed on commercial fleets such as buses, due to their predictable mobility patterns (e.g., schedules and driving routes). Accordingly, the supply of VFNs depends on the mobility pattern of buses, while the supply of CFNs depends on the deployment of cellular BSs. Based on real-world bus schedules, we divide a target area into clusters and map bus journeys using a spatiotemporal availability matrix. Here, a bus journey defines the driving route as well as the time of day when the trip starts. The same journeys are typically repeated on a daily basis during weekdays, and on a weekly basis during weekends. The same bus journeys may be served by different buses on different days.

Our cost minimization module produces the deployment plan of CFNs, the selection of bus journeys, and the minimized operational cost. In this module, we assume that all

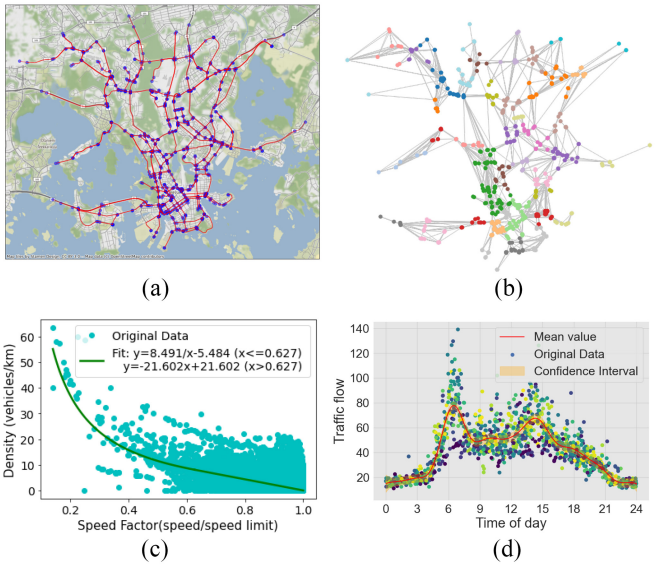

Fig. 3. Process of spatiotemporal traffic modeling. (a) Road network of our study area, where each red line represents a road segment, and each blue dot represents a road intersection. (b) Clusters of road segments using k-means, colors are used to distinguish clusters. (c) Traffic density versus normalized speed based on traffic flow theory. (d) Example of traffic flow versus time of day during the weekdays.

the busses carry VFNs in the study area. However, this may cause an oversupply of VFNs. Our final step is to run the bus scheduling module to identify a minimal subset of buses to cover the selected bus journeys for VFN deployment. The bus journeys that belong to the same bus line and have sufficient turnaround time are chained together to ensure the same bus can cover different trips. Therefore, we can minimize the installation cost of VFNs.

V. Demand Estimation

In this section, we introduce our demand estimation approach. We first establish the spatio-temporal traffic model to understand how vehicular traffic varies over time and among locations. Additionally, we illustrate the application profiling process to determine the consumption pattern of the vehicular applications. Finally, with these inputs, we formulate the demand estimation problem.

A. Spatial–Temporal Traffic Modeling

Fig. 3 shows the process of spatiotemporal traffic modeling. We use k-means clustering to group the road segments into clusters based on the road network. Two types of traffic data sets are used to derive the traffic density according to the traffic flow theory [33]. Finally, we model the daily traffic flow as a distribution of time of day using Gaussian process regression.

1) Road Network: We use a graph $G = (V, E)$ to show a road network, where each vertex V represents a road segment, and each edge E indicates a road intersection. The road segment is the basic unit of the road network, and the road intersections represent the topological relationship of the road segments.

2) Road Segment Clustering: We group the road segments into clusters based on their geographical relationship, and the traffic flow is accumulated in each cluster to estimate the demand. At each time slot, the CFNs and VFNs will serve the client vehicles within the same cluster. We use k-means for clustering the road segments, and there should be at least one BS inside each cluster.

3) Traffic Density Derivation: According to the traffic flow theory, the basic variables of traffic flow are the average speed, flow rate, and traffic density. If we know any two of these variables, we can derive the remaining one [33]. There are usually two types of traffic data sets. One type is the speed data set, which samples the average speed of the vehicles on each road segment at each time slot. The other type is the flow rate data set, which records the number of vehicles that pass through a site (e.g., a traffic monitoring station) during a time interval. We calculate the traffic density of each road lane by dividing the flow rate from the latter data set by the average speed in the former data set. Then, we apply piecewise regression to get the relationship between the traffic density and the normalized speed (i.e., the ratio of average speed and the speed limit). With this relationship, we can estimate the traffic density of all the road lanes in the city based on their normalized speed. We then calculate the spatiotemporal traffic flow (i.e., the number of vehicles on each road segment at each time slot) by multiplying the traffic density with the number of lanes and the length of the road segment.

4) Traffic Flow Modeling: We use Gaussian process regression to model the daily traffic flow in terms of vehicles per cluster, with the predicted mean and variance functions. Assume the mean function and variance function in a cluster have the value $\bar{X}$ and $\hat{\sigma}$ at a certain time. To get the upper bound of the confidence interval, we use $\bar{X} + \beta \times \hat{\sigma}$ to denote the spatiotemporal traffic flow, where β represents the coefficient of the variance in the confidence interval. The traffic flow is modeled separately during weekdays and weekends due to different time of day patterns.

B. Vehicular Application Profiling

Apart from the vehicular traffic, the demand of the fog computing system also depends on the resource consumption of the vehicular applications, which is reflected in the CPU and GPU consumption [8]. The vehicular applications are containerized into Docker [34] Images, and we design a set of benchmark testing for each containerized application. Algorithm 1 presents our application benchmark testing algorithm. After getting the vehicular application profiles, nonlinear least-squares regression is used to map the mathematical relationship among the CPU usage, GPU usage, and latency. Then, we find the CPU and GPU consumption under different latency requirements based on the mathematical relationship.

C. Formulation of Demand Estimation

The demand estimation problem aims to find the minimum amount of computing capacity required in each cluster at each time slot to meet the computing tasks generated by the users of vehicular applications. The computing capacity is represented

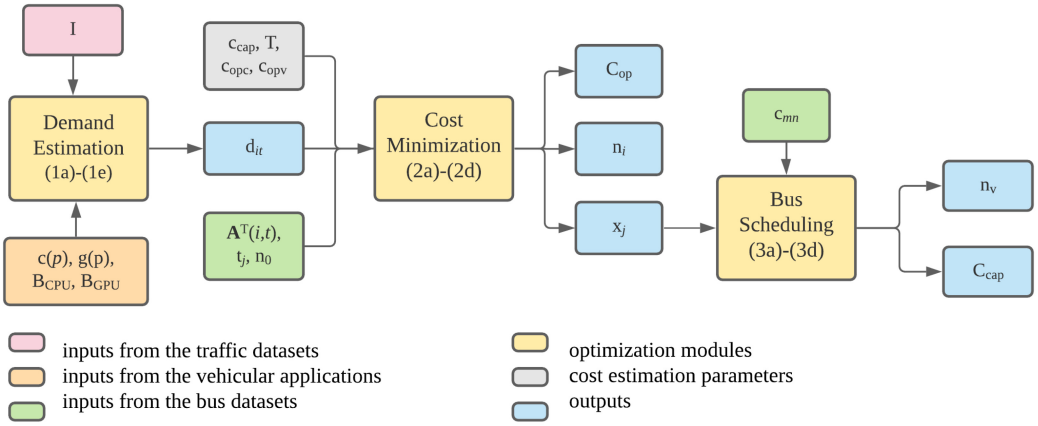

Fig. 4. Flowchart of the capacity planning model.

Algorithm 1: Application Benchmark Testing Algorithm

Input: computing latency requirement $r_{compute}$
Output: mean of frame latency $\mu_{compute}$, standard deviation of frame latency $\sigma_{compute}$, mean of CPU usage μ_{cpu}, mean of GPU usage μ_{gpu}

```
i = 1;
while μ_compute ≤ r_compute do
    start the docker service of i replicas of application;
    while service is running do
        record the frame latency;
        record the CPU and GPU usage per second;
    end
    calculate μ_compute, σ_compute, μ_cpu, μ_gpu;
    i = i + 1;
end
```

as the number of FNs with fixed unit size. I represents the set of computing tasks. Assume each user will keep one active computing task at each time slot, then the number of the computing task is equal to the number of users. This is denoted as $|I| = n$, where n is the number of users, and $|.|$ represents the cardinality of the set. The CPU and GPU consumption of each computing task p are represented by $c(p)$ and $g(p)$, respectively. The CPU and GPU consumption depend on the vehicular application type and the latency requirement. We assume the latency requirement is universal for the users at each time. We also need to know the maximum capacity of the CPU and GPU according to the configurations of the FNs, which are given by B_{CPU} and B_{GPU}, respectively. In this work, we assume that all the FNs have the same CPU and GPU configurations.

The demand estimation problem is given in (1a)–(1e). Our objective function is (1a), where $\lceil . \rceil$ represents the ceiling function. The object function reflects that an FN is needed if at least one computing task is assigned to it, and the function minimizes the required number of FNs to serve all the computing tasks generated from the users. Constraints (1b) and (1c) are the *CPU configuration constraint* and *GPU configuration constraint*, respectively, which prevent the computing tasks assigned to each FN from exceeding the CPU and GPU configuration of the FN. Constraint (1d) is the *nonrepetitive assignment constraint* ensuring that each computing task is not assigned to multiple FNs. Finally, constraint (1e) defines the binary decision variable x_{pq}, which indicates if the task p is assigned to the FN q.

Since the objective of the above problem is to determine the minimum number of FNs, the problem can be solved in polynomial time using a heuristic algorithm formulated as follows. We start by picking a random FN and assigning the tasks to it until it is full. We repeat this process of selecting random FNs and assigning tasks until we have allocated all the tasks or all the FNs are full

$$\min_{x_{pq}} \quad \sum_{q=1}^{n} \left\lceil \frac{\sum_{p \in I} x_{pq}}{n} \right\rceil \tag{1a}$$

$$\text{s.t.} \quad \sum_{p \in I} c(p) x_{pq} \leq B_{\text{CPU}} \quad \forall q \in (1, 2, \ldots, n) \tag{1b}$$

$$\sum_{p \in I} g(p) x_{pq} \leq B_{\text{GPU}} \quad \forall q \in (1, 2, \ldots, n) \tag{1c}$$

$$\sum_{q=1}^{n} x_{pq} = 1 \quad \forall p \in I \tag{1d}$$

$$x_{pq} \in (0, 1) \quad \forall p \in I \quad \forall q \in (1, 2, \ldots, n). \tag{1e}$$

VI. Cost-Optimal Fog Nodes Deployment

This section focuses on how to deploy CFNs and VFNs to fulfill the estimated computing resource demand generated from the vehicular applications. Different from CFNs, which are located at stationary cellular BSs, the locations of VFNs depend on the schedules and driving routes of the carriers, which are buses in this case. Therefore, we start by estimating the availability of VFNs based on the mobility pattern of buses in Section VI-A. Then, we move to the problem of minimizing the operational and installation costs through the optimal distribution of computing capacity on the cellular BSs and buses. In Section VI-B, we present the algorithm for selecting the bus journeys to be served by buses carrying VFNs, with the aim of minimizing the overall costs. Instead of installing VFNs on all buses, Section VI-C focuses on scheduling the routes of a subset of buses to cover the bus journeys selected in Section VI-B, in order to further minimize the installation costs. Fig. 4 outlines the architecture of the capacity planning model.

A. Bus Mobility Pattern

We describe a bus journey m with the bus line l_m, direction r_m, departure time dp_m, and the one-way travel time tr_m. The bus line and direction together determine the driving route of the bus journey. During a journey, a bus may go through several road segment clusters.

1) Spatiotemporal Availability Matrix: To model the mobility of buses, we use a matrix to describe the spatiotemporal distribution of bus journeys. For cluster i and time slot t, the spatiotemporal availability matrix $\mathbf{A}(i,t)$ is given as a vector of size $(u \times 1)$, where u is the number of bus journeys in the study area. Each element in the vector is a binary value indicating the availability of the bus taking the journey. The value is 1 if the bus taking each journey is traveling in the cluster in question; otherwise, the value is 0. Accumulating the time slots with nonzero elements for each journey in the study area, we get the duration of each journey dur_j.

If a VFN is placed on a bus, the VFN service becomes available along the bus journeys taken by the bus. When estimating the spatial distribution of VFNs in each time slot, we go through such bus journeys and calculate the communication range of each VFN. Each VFN is associated with the nearest road segment cluster within its communication range. Tasks generated within a cluster are supposed to be executed only on the associated VFNs.

2) Adjacency of Bus Journeys: In addition, we use a parameter to indicate the adjacency relationship between each pair of journeys m and n. More precisely, it shows whether journey n can be covered after journey m by the same bus. A journey can be adjacent with another if both journeys belong to the same bus line, and the departure time of the upcoming journey is later than the arrival time of the last journey. Therefore, we define the adjacency between two journeys m and n as

$$c_{mn} = \begin{cases} 1, & \text{if } l_m = l_n \text{ and } r_m = r_n \\ \text{and } dp_m + 2tr_m \le dp_n & \\ 1, & \text{if } l_m = l_n \text{ and } r_m \neq r_n \\ \text{and } dp_m + tr_m \le dp_n & \\ -\infty, & \text{otherwise.} \end{cases}$$

B. Cost Minimization

The cost minimization problem aims to minimize the overall installation and operational costs of the VFC system. The installation cost per FN is represented by c_{cap}, and the overall installation cost in this problem is represented by C'_{cap}, where $'$ indicates that we can further minimize the installation cost by bus scheduling. The installation cost includes the charges of purchasing and installing the FNs, which are paid only once. The overall number of buses in the region is denoted by n_0.

The operational cost of CFN and VFN per unit time is presented by c_{opc} and c_{opv}, respectively, and the overall operational cost is denoted by C_{op}. The operational cost includes at least the rent, the power consumption fee (e.g., fuel and electricity), and regular maintenance; the cost proportionally increases with the operating time. To calculate the operational cost, we need to specify the capacity planning horizon T in days. We also need to record the duration dur_j of each bus journey $j(j \in U)$ in the study area, where U is the set of bus journeys, and u denotes the number of the bus journeys.

The inputs also include the spatiotemporal demand estimation, represented by the demand d_{it} in cluster $i(i \in S)$ at time slot $t(t \in W)$. S is the set of clusters, and W is the set of time slots in a day. We also need the spatiotemporal availability matrix $\mathbf{A}(it)$ from the bus data set, represented by a vector of size $(u \times 1)$ in cluster i at time slot t.

Assuming all the buses in the area carry the VFNs, we can write the installation cost as

$$C'_{\text{cap}} = c_{\text{cap}} \left(\sum_{i=1}^{S} n_i + n_0 \right).$$

The operational cost can be written as

$$C_{\text{op}} = T \left(c_{\text{opc}} \times W \sum_{i=1}^{S} n_i + c_{\text{opv}} \times \sum_{j=1}^{U} \text{dur}_j x_j \right).$$

The cost minimization problem is given in (2a)–(2d). Our objective function, cf. (2a), minimizes the overall installation and operational costs. Constraint (2b) is the *spatiotemporal capacity constraint*, which ensures that the capacity provided by the CFNs and VFNs is sufficient for the estimated demand for each cluster at each time slot. To plan the CFN and VFN deployment, we define two decision variables in constraints (2c) and (2d), respectively. The first decision variable n_i is an integer variable to indicate the number of CFNs in cluster i. The second decision variable x_j is a binary variable to indicate whether the bus journey j is selected to serve as the VFNs. The vector form of all the vehicular FNs decision is $\mathbf{X}$, with the size of $(u \times 1)$

$$\min_{n_i, x_j} \quad C'_{\text{cap}} + C_{\text{op}} \tag{2a}$$

$$\text{s.t.} \quad n_i + \mathbf{A}^T(i,t)\mathbf{X} \ge d_{it} \quad \forall i \in S \quad \forall t \in W \tag{2b}$$

$$n_i \in \mathbb{Z}^+ \quad \forall i \in S \tag{2c}$$

$$x_j \in (0,1) \quad \forall j \in U. \tag{2d}$$

C. Bus Scheduling

In real life, the route of each bus line is fixed, and several buses may serve the same bus line. Depending on the timetable, there may be several bus journeys between the origin and destination of each route. We define a bus journey as the bus line, bus route (including the origin and destination), and departure time. We consider these aspects when deciding which bus will be responsible for each bus journey. Only when two journeys are within the same bus line, and the departure time interval between the two trips is greater than the turnaround time between their routes, can the two journeys be covered by the same bus in the time order. We repeat this process until all the selected bus journeys are covered.

The bus scheduling problem aims to find the minimum number of buses to cover the selected bus journeys, which would further minimize the installation cost. The minimal decomposition model [35] is used for the problem. The set of selected journeys is denoted by J. The number of the selected journeys $|J| = k$, where $|.|$ represents the cardinality of the set. To

schedule the buses, we also need the adjacency relationship c_{mn} for each journey pair.

We present our bus scheduling problem in (3a)–(3d). The objective function is (3a), based on the Dilworth theorem of partially ordered sets [35]. It minimizes the number of buses to cover the selected journeys (i.e., the minimum decomposition of the bus journey set J [35]). Constraints (3b) and (3c) are the *nonrepetitive scheduling constraints* in two directions. They guarantee that each bus journey is either covered by an individual bus or covered in a sequence of bus journeys. Finally, constraint (3d) defines the binary decision variable, which indicates whether a bus is scheduled to cover journey n after journey m

$$\min_{b_{mn}} \quad k - \sum_{m\in J}\sum_{n\in J} c_{mn} b_{mn} \tag{3a}$$

$$\text{s.t.} \quad \sum_{n\in J} b_{mn} \le 1 \quad \forall m \in J \tag{3b}$$

$$\sum_{m\in J} b_{mn} \le 1 \quad \forall n \in J \tag{3c}$$

$$b_{mn} \in (0,1) \quad \forall m \in J \quad \forall n \in J. \tag{3d}$$

With the minimum number of buses, we can write the minimized installation cost as

$$C_{\text{cap}} = c_{\text{cap}}\left(\sum_{i=1}^{S} n_i + n_{\text{v}}\right) \tag{4}$$

where n_{v} equals to the value of the objective function in the bus scheduling problem.

VII. Experimental Setup

Real-world data sets and applications are used to validate the capacity planning framework. This section introduces the data sets, applications, and simulation settings.

1) Helsinki Speed Data Set: We extracted from HERE [36] a map that covers 869 road segments in Helsinki, ranging from latitude 60.222306, longitude 24.858754 to latitude 60.142211, longitude 24.993980. Each road segment is associated with a set of sequential coordinates, including the start and end intersections. We created the Helsinki road network based on this map. We then created a Helsinki speed data set that includes speed samples of all the road segments, collected through HERE Traffic API [37]. The speed samples were collected once in a minute from January to February 2020.

2) Helsinki Flow Rate Data Set: The Helsinki flow rate data set was updated daily by the traffic monitoring system (TMS) of the Finnish Transport Agency [38]. The traffic monitoring stations were located at all the major roads in Finland, and the flow rates of these roads in the same study area were sampled during the same time as the Helsinki speed data set. Combining these two data sets, we can establish the spatiotemporal traffic models.

3) Helsinki Bus Position Data Set: The Helsinki bus position data set was collected using Helsinki regional transportation (HSL) high-frequency positioning (HFP) API [39]. All the buses in Helsinki update their status online, including their bus line, vehicle id, direction, departure time, and real-time position every second. The bus position data were downsampled to one sample per minute to be aligned with the former two data sets. From the bus position data set, we can model the bus mobility patterns.

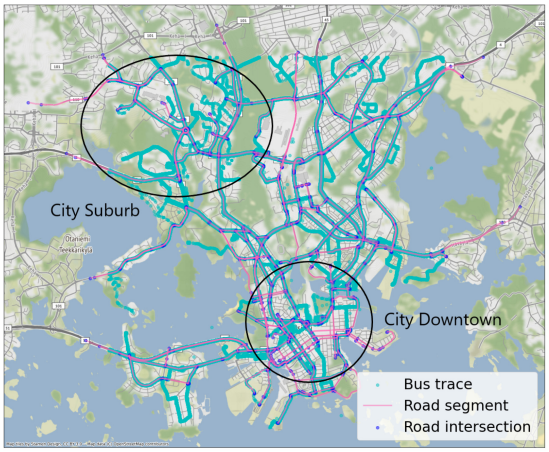

Fig. 5. Helsinki city map that covers 869 road segments and 9421 bus traces.

4) Helsinki Bus Timetable: The Helsinki bus timetable is updated seasonally by HSL general transit feed specification (GTFS) [40]. It records the departure time and arrival time of all the bus journeys at every bus stop. From the bus timetable, we get the driving routes and adjacency relationship of the bus journeys.

5) City Downtown and City Suburb: As shown in Fig. 5, we classified some clusters into two areas, namely, the downtown and suburb area, depending on their geographical information. Each of these areas consists of six clusters. The city downtown covers the city center, a commercial region, and a high-density residential area. The city suburb covers highways, natural parks, and a low-density residential area.

A. Vehicular Applications

To exemplify the compute-intensive and latency-sensitive vehicular applications, we chose four computing tasks for testing, including object detection, semantic segmentation, lane detection, and video transcoding. All of them are data-intensive applications, with a driving footage video with 1280×720 resolution and 25 fps frame rate as the input. Assuming the memory is sufficient for all the applications, they have different CPU and GPU consumption patterns described as follows.

1) Object Detection: The object detection application is implemented through YOLOv5s [41] trained on the COCO data set [42]. The algorithm divides images into a grid system, where each cell in the grid is responsible for detecting objects within itself. It represents the low-CPU and medium-GPU applications [shown in Fig. 7(a) and (b)].

2) Semantic Segmentation: The semantic segmentation application is implemented through image segmentation Keras [43] with the VGG-UNET model and trained on the Cityscapes data set [44]. The algorithm classifies each pixel in an image from a predefined set of classes using a fully convolutional network. It is a representative of the medium-CPU and high-GPU applications [shown in Fig. 7(c) and (d)].

TABLE II
SIMULATION SETTINGS

	Simulation 1	Simulation 2	Unit
c_{cap}	1000	500, 1000, 1500, 2000	MU/device
c_{opc}	0.02	0.02	MU/minute
α_{op}	0.5, 1.0, 1.5, 2.0	1.0	/
T	1.3	0.78, 1.04, 1.3, 1.56, 1.82	10e3day
r_{comp}	250, 150, 100	100	millisecond
β	3		/
p_{task}	1:1:1:1		/
W	1440		minute
S	6 in downtown, 6 in suburb		cluster
U	5189 in downtown, 5853 in suburb		journey
n_0	543 in downtown, 603 in suburb		bus

3) Lane Detection: The lane detection application is implemented by OpenCV [45] in a Python environment. It includes image processing techniques, such as color selection, canny edge detection, region of interest selection, and Hough transform line detection. It is a representative of the medium-CPU applications [shown in Fig. 7(e)].

4) Video Transcoding: The video transcoding application is implemented by HandBrake video transcoder [46] with x265 video encoder and mp4 container. It contains both the process of decoding and encoding, which represent the high-CPU applications [shown in Fig. 7(f)].

We used an Intel Core i7-7700K CPU with eight threads that can run in parallel. Assuming the computing capacity for each thread is 100%, the capacity of the CPU $B_{\mathbf{CPU}} = 800\%$. We use a NVIDIA GeForce RTX 2080 Ti GPU. The capacity of the GPU $B_{\mathbf{GPU}} = 100\%$.

B. Simulation Setup

To validate the functionality of our model, we consider three deployment strategies in the experiment, as detailed as follows.

1) *Deploying Strategy (DS):* Deploying FNs on both cellular BSs and buses, and installing VFNs on a minimum number of buses that can cover the selected bus journeys (i.e., where the number of VFNs equals to n_v).
2) *Comparing Strategy—CFNs Only (CP-CO):* Deploying FNs only on cellular BSs.
3) *Comparing Strategy—All Buses (CP-AB):* Deploying FNs on both cellular BSs and buses and installing VFNs on all the buses in the study area (i.e., where the number of VFNs equals n_0).

Among these three strategies, our framework adopts *DS*, while the other two strategies are used for comparison. *CP-CO* corresponds to the traditional stationary FN deployment model, and *CP-AB* corresponds to the output of the capacity planning model without bus scheduling module. We control the overall demand that will be fulfilled by the FNs as equal in all of the strategies (i.e., all of them are from the outputs of the demand estimation module). We compare the three strategies in terms of cost efficiency.

Table II details the two sets of simulations. The given costs are purely designed for comparison purposes and given in monetary units (MUs). However, we investigate the impacts of the relative operational costs in Section VI-B. In both simulations, the confidence interval in traffic modeling is selected as three times of standard deviation of the estimated traffic flow. Furthermore, the probability of selecting each type of computing task is set as equal for each user.

The first simulation scenario analyzes the impacts of latency requirement and operational cost on the deployment plans of CFNs and VFNs (see Section IX-A). To analyze the implications of latency requirement, we change the computing latency requirements from 250 to 150 and 100 ms. The operational costs include at least rent, electric expenses, and maintenance fees. The pricing level of each cost varies over time and with the location [47]. For generalization, we have defined the relative operational cost as the ratio of the unit operational cost of CFNs to that of VFNs, denoted as

$$\alpha_{\text{op}} = c_{\text{opc}}/c_{\text{opv}}. \tag{5}$$

To analyze the implications of the relative operational cost, we change it from 0.5 to 2.0 while keeping other parameters as constants. For example, the ratio of 1.0 represents the scenario where operational costs of CFN and VFN are equal (i.e., for both, it is 0.02 MU/min). The ratio of 0.5 represents the scenario where the operational cost of CFN is 50% lower than VFN (i.e., for CFN, it is 0.02 MU/min, and for VFN, it is 0.04 MU/min).

The second simulation scenario compares the installation and operational costs among the three deployment strategies under different unit installation costs and operation time (see Section IX-B). In this set of simulations, while keeping other parameters as constants and setting the relative operational cost to 1.0, we change the unit installation cost from 500 to 2000 MU/device and change operational time from 780 to 1820 days (i.e., approximate working days from three to seven years). To estimate the overall costs in the long term, we assume that the vehicular traffic and bus mobility patterns remain unchanged during the operational time. Considering they may change in the real-life scenario, we will further discuss this topic in Section X-A.

We compare the deployment decision and the cost estimation in the city downtown versus suburb, on weekdays versus weekends (see Sections IX-A and IX-B). For the convenience of comparison, we set the time range of the weekend models equal to the weekday models. We formulate our framework in Fig. 4 as separate optimization models. Among these models, we solve the demand estimation module using the heuristic method detailed in Section V-C. The remaining modules are developed in Python 3.8 and solved using the Gurobi [48] solver.

VIII. FUNCTIONALITY EVALUATION

In this section, we evaluate the functionality of our framework through a real-world simulation, including traffic modeling, application profiling, bus journey selection, and bus scheduling.

A. Traffic Models

Fig. 6 shows the spatiotemporal distribution of the traffic flow, where each color represents a cluster in the area. Despite having a smaller geographical coverage, the downtown area

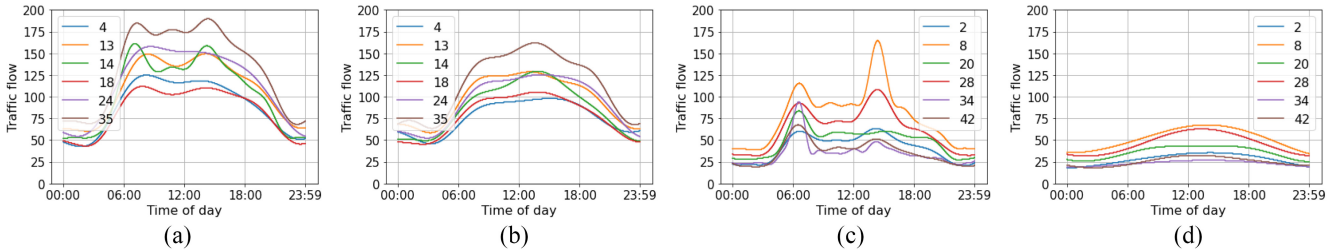

Fig. 6. Spatiotemporal traffic models, where each color represents a cluster labeled in the legend. The traffic flow is represented by the number of vehicles in each cluster shown on the *y*-axis. (a) Weekdays in city downtown. (b) Weekends in city downtown. (c) Weekdays in city suburb. (d) Weekends in city suburb.

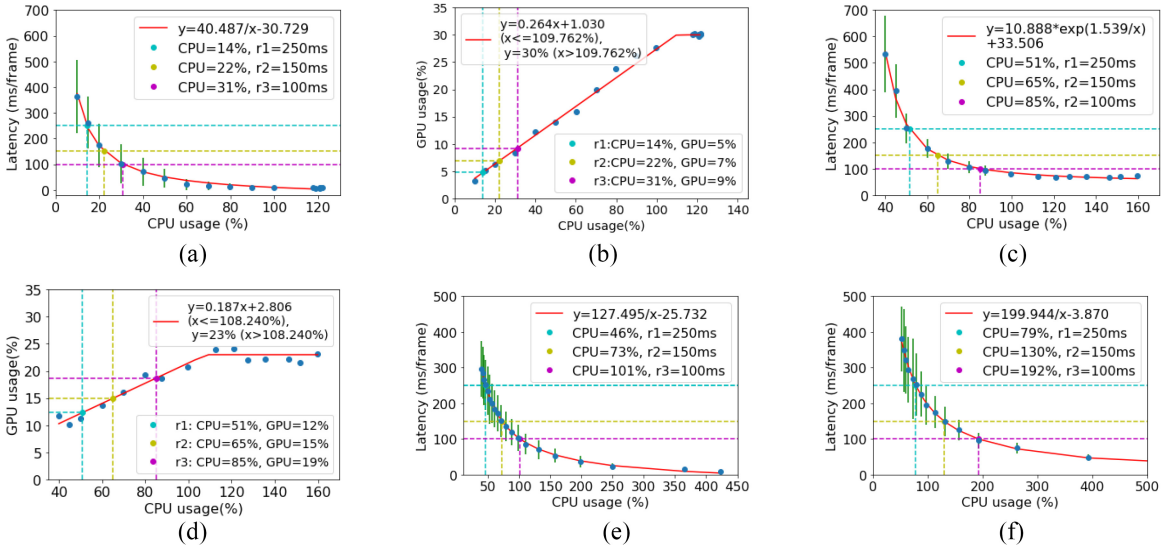

Fig. 7. Vehicular application profiles, where the blue dots and green bars represent the mean and standard deviation of frame latency. The red curves represent the regression results, and the dashed lines represent the computing latency requirements. (a) Object detection latency versus CPU. (b) Object detection GPU versus CPU. (c) Segmentation latency versus CPU. (d) Segmentation GPU versus CPU. (e) Lane detection latency versus CPU. (f) Video transcoding latency versus CPU.

accommodates a larger traffic volume than the suburban area. The traffic density is higher in the city downtown, especially during the weekdays. The traffic flow has different time of day patterns between weekdays and weekends. During the weekdays [Fig. 6(a) and (c)], there are usually two peaks in the traffic flow corresponding to the morning and afternoon commuting hours. However, on the weekends [Fig. 6(b) and (d)], we usually observe one peak around noon. The traffic flow also shows the difference between the downtown and suburb areas throughout the day. In the downtown area [Fig. 6(a) and (b)], the traffic flow remains at high throughout in the daytime, while in the suburb area [Fig. 6(c) and (d)], it is low apart from the peak hours.

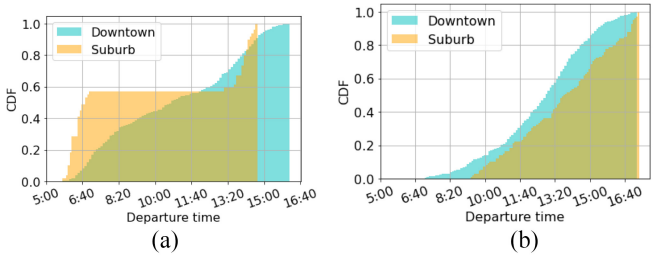

Fig. 8. Cumulative distribution function (CDF) of departure time of selected bus journeys with different traffic patterns. (a) Weekdays (downtown 375 journeys versus suburb 48 journeys). (b) Weekends (downtown 410 journeys versus suburb 111 journeys).

B. Application Profiles

Fig. 7 shows the application profiles, represented by the latency versus the CPU and GPU consumption. We can observe that the resource consumption patterns of the four application is consistent with the description in Section VII-A. The figures also show that more stringent latency requirement necessitates higher CPU and GPU usage until they have reached the maximum value.

C. Bus Journey Selection

The cost minimization module minimizes the operational cost while deciding which bus journey will be taken care of by which bus carrying VFN. We compare the selected bus journeys among different traffic patterns in terms of their departure time. From Fig. 8, we can see that during the weekdays, the departure times of the selected bus journeys are mainly concentrated in the morning and afternoon peak hours in the

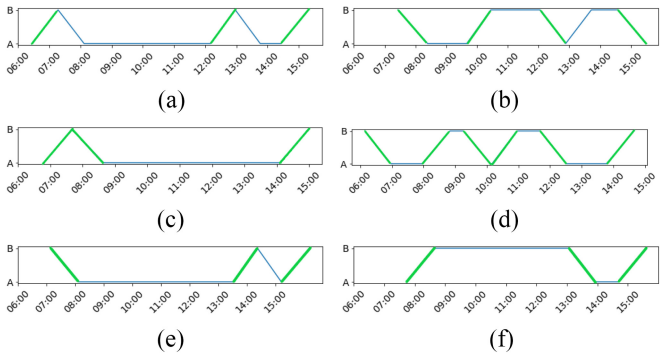

Fig. 9. Schedules of the six VFNs from bus line No. 322 in Helsinki, where A and B are the departure stops in two directions. The green lines represent the schedules of the 22 selected journeys, and the blue lines represent the turn-around time between the journeys. (a) Schedule of VFN ID 1. (b) Schedule of VFN ID 2. (c) Schedule of VFN ID 3. (d) Schedule of VFN ID 4. (e) Schedule of VFN ID 5. (f) Schedule of VFN ID 6.

suburb, while the departure times in the downtown have a more even distribution from the morning to the afternoon. During the weekends, the departure times of the selected bus journeys are concentrated around noon in both the city downtown and city suburb, while the departure times in the downtown have a wider distribution. The results show that bus journey selection is consistent with the spatiotemporal traffic patterns since the departure times match the moments when the traffic flow increases in the traffic models.

D. Bus Scheduling Plan

In real life, a bus line between Station A and Station B is operated by a certain number of buses. Each bus journey refers to a trip from Station A to Station B, or *vice versa*. Throughout this article, we assume that the buses, no matter whether they carry FNs or not, do not change their routes and timetables during the capacity planning horizon. After the bus journeys are chosen, the bus scheduling module decides which bus will be assigned to each journey. During the simulation, we schedule the buses according to the 2020 winter timetables from HSL GTFS [40]. We demonstrate an exemplary scheduling plan of bus line No. 322 in Helsinki. There are 25 buses serving this line and in total 79 bus journeys a day. Among them, 22 bus journeys are selected in the cost minimization module. According to the bus scheduling result, six buses will install VFNs. From Fig. 9, we can see the six buses carrying the VFNs can cover the 22 selected journeys, and there is no conflict in their timetables. Following this procedure, a subset of buses is chosen for FN deployment instead of all buses in the region, which greatly reduces the installation cost of VFNs.

IX. Capacity Plan, Cost Efficiency, Service Provision, and Impact Analysis

This section presents the results of the capacity plan and cost estimation. It analyzes the impacts of the traffic patterns, latency requirements, and relative operational cost on the capacity plan. It also investigates the effects of changing the traffic patterns and cost estimation parameters on the cost-efficiency analysis. Furthermore, this section compares different capacity planning strategies in terms of service rate based on a microscopic VFC simulation.

TABLE III
Comparison of FN Distribution on Weekdays Using *DS*, *CP-CO*, and *CP-AB*, Where # Represents the Cluster ID

	CFN						VFN
Downtown	**#4**	**#13**	**#14**	**#18**	**#24**	**#35**	**region**
DS	16	20	21	14	21	22	146
CP-CO	17	20	22	15	21	25	0
CP-AB	16	20	21	14	21	22	543
Suburb	**#2**	**#8**	**#20**	**#28**	**#34**	**#42**	**region**
DS	9	20	10	14	13	9	45
CP-CO	9	22	11	14	13	9	0
CP-AB	9	20	10	14	13	9	603

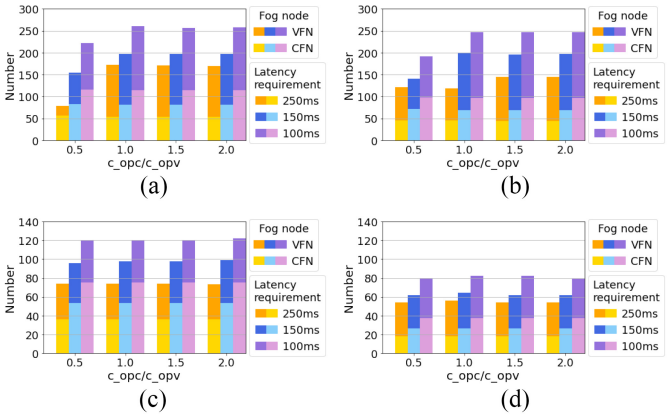

Fig. 10. Effects of latency requirement, relative operational cost, and traffic models on FN deployment decisions. The maximum number (percentage) of VFNs is 144 (56%), 151 (61%), 47 (39%), and 42 (53%), respectively, in the four subfigures. (a) Weekdays in city downtown. (b) Weekends in city downtown. (c) Weekdays in city suburb. (d) Weekends in city suburb.

A. Capacity Plan

In Section VII-B, we describe the three different deployment strategies, namely, *DS*, *CP-CO*, and *CP-AB*. Table III compares the resulting deployment plans on weekdays. It lists the number of CFNs in each cluster and the number of VFNs that travel through the target region. The results are obtained from the experiments where the latency constraint is set to 100 ms while the relative operational cost is set to 1.0. *DS* requires fewer CFNs than *CP-CO* and fewer VFNs than *CP-AB*, given the same QoS requirements. Compared with *CP-CO*, *DS* includes more VFNs. Section IX-B details the comparison of overall costs.

Fig. 10 shows the impacts of latency requirement, relative operational cost, and traffic pattern on the FN deployment using *DS*. In the figure, the *x*-axis is the relative operational cost, whereas the *y*-axis shows the number of FNs. The number of CFNs and VFNs is stacked together in each bar, and the three bars in parallel represent the three settings of latency requirement, respectively.

Impacts of Latency Requirement: Given a relative operational cost, if we compare the numbers of FNs under different latency requirements, we can find that more computing

resources or FNs are needed to fulfill stricter latency requirements.

Impacts of Relative Operational Cost: According to Fig. 10(a) and (b), when the relative operational cost increases, the percentage of VFNs in the downtown area will increase until it reaches the maximum value. In other words, when the operational cost of VFNs becomes relatively low, the model tends to select more bus journeys. We can see in Fig. 8 that the selected bus journeys are concentrated at the times and places where the traffic flow increases. This indicates that at other times and places, employing CFNs will be a more cost-efficient solution than employing VFNs. The selection of the bus journeys will stop at the saturation point when the departure time and routes of the remaining journeys are not within the high traffic hours and regions, and the proportion of the VFNs will stop increasing. The maximum values and percentages of VFNs can be found in the caption of Fig. 10. According to Fig. 10(c) and (d), the percentage of VFNs does not change with the relative operational cost in the suburb area. This is because the selection of the buses has already reached the saturation point at the first setting.

Impacts of Traffic Pattern: Based on Figs. 6 and 10, we can see that traffic density is one of the main decision factors in the capacity plan. The number of FNs increases with the traffic density because the FNs are necessary at times and places with higher demand. From Figs. 8 and 10, we can see that another impacting factor is the hourly variation in traffic flow. When the traffic flow increases significantly during peak hours, more bus journeys are selected, and more VFNs are deployed. The above findings also hold true for different clusters within the city downtown and city suburb. From Table III, we can see the number of CFNs is higher in the clusters with higher traffic density (e.g., cluster #35, #24 in the city downtown and cluster #8 in city suburb). The difference of the numbers of CFNs using *DS* and *CP-CO* is higher in the clusters with higher daily variation (e.g., cluster #35, #14 in the city downtown and cluster #8, #20 in the city suburb) because more CFNs will be substituted by the VFNs.

B. Cost Efficiency

Fig. 11 shows the comparison of the installation and operational costs among the three strategies. It then compares *DS* and *CP-CO* in terms of the overall cost and the saving potential of *DS* with different traffic patterns.

Installation Cost: Fig. 11(a) shows how the installation cost changes with the unit installation cost. The installation cost of *CP-CO* is the lowest and that of *CP-AB* is the highest. The cost difference between the strategies increases when the unit installation cost becomes higher. Compared to *CP-CO*, *DS* has a higher installation cost. This is because the CFNs provide stable computing services at their corresponding clusters, while VFNs can only provide the computing services as they pass through the clusters along their driving routes. To substitute the CFN, much more VFNs are required to present at the times and places with higher demand. Compared to *CP-AB*, *DS* significantly reduces the installation cost. Therefore, it is more feasible to install the VFNs on part of the buses and schedule them.

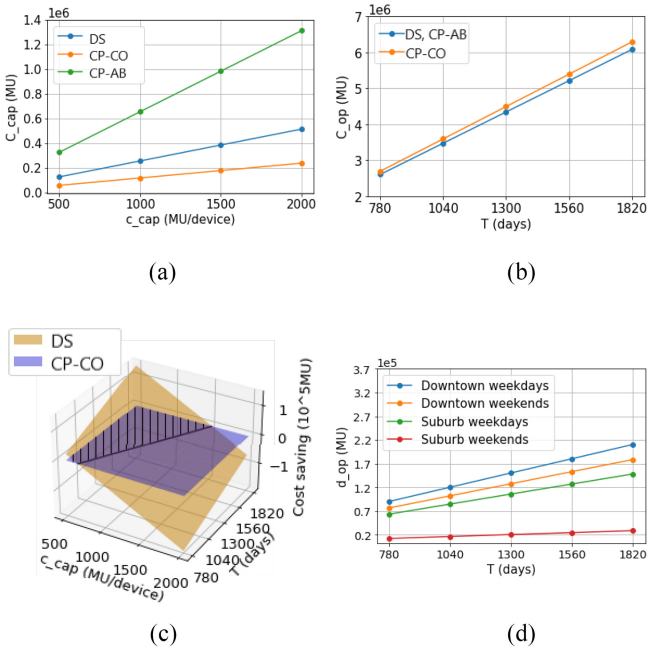

Fig. 11. Cost analysis of different deployment strategies. (a) Installation cost versus unit installation cost using *DS, CP-CO, and CP-AB*. (b) Operational cost versus operational time using *DS, CP-CO, and CP-AB*. (c) Comparison of overall cost using *DS* and *CP-CO*. (d) Saving of operational cost using *DS* with different traffic patterns.

Operational Cost: Fig. 11(b) shows how the operational cost changes with the operation time. The operational cost of *DS* and *CP-AB* coincides with each other since they follow the same process of bus journey selection to get the same output. Compared to *CP-CO*, *DS* has a lower operational cost, and the cost difference increases when the operation time becomes longer. The deployment of VFNs reduces the idle state of the computing resources, thus saving the operational cost compared to *CP-CO*.

Overall Cost: Fig. 11(c) shows the overall cost using *DS* and *CP-CO*. Since *CP-AB* has much higher cost compared to the other strategies, it is not plotted. In the figure, the x-axis is the unit installation cost, and the y-axis is the operation time. The cost estimation of *CP-CO* is given as the blue plane of $z = 0$, and the cost estimation of *DS* is shown as the orange plane. In the shaded area, the z-value of the orange plane is positive, meaning that *DS* is more cost effective than *CP-CO*. When the unit installation cost becomes lower, or when the operational time becomes longer, *DS* will have higher potential for cost-saving compared to *CP-CO*. Therefore, we can conclude that VFC is more suitable for the cases where the operational cost has a larger weight than the installation cost.

Long-Term Saving of Operational Cost: From the above analysis, we already know that the deployment of VFNs saves the operational cost at the expense of adding installation cost. To estimate the saving of operational costs between different times and places, we compared them in Fig. 11(d). The results show the saving of operational cost is larger in the

city downtown than in the city suburb, and larger during the weekdays than the weekends. Additionally, when the operational time becomes longer, the saving of operational cost becomes more significant. Therefore, we can conclude that the long-term saving potential of operational cost is greater in the times and areas with higher traffic density and greater daily variation because more VFNs will be deployed.

TABLE IV
SERVICE RATES OF DIFFERENT DEPLOYMENT STRATEGIES UNDER VARIOUS LATENCY REQUIREMENTS IN PEAK AND OFF-PEAK SCENARIOS

	Off-peak Scenario			**Peak Scenario**		
$r_{network}$	50 ms	100 ms	150 ms	50 ms	100 ms	150 ms
DS	84.0%	92.0%	98.0%	69.3%	88.7%	95.3%
CP-COS	80.0%	92.0%	98.0%	60.0%	87.3%	94.7%
CP-COE	88.0%	92.0%	98.0%	75.3%	92.7%	96.0%

C. Service Provision

In this section, we analyze the actual service rate, i.e., the percentage of users that can be served with a given deployment strategy. The network dynamics, such as the signal-to-interference-plus-noise ratio (SINR), available bandwidth, and communication range, play a major role in the actual service rate of a deployment strategy. We use a VFC simulator [49] to measure the service rates of different deployment strategies.

We assume that the execution time for vehicular tasks is negligibly short. Therefore, we model the latency constraints to reflect only the network latency. The time is discretized and divided into time slots, which is denoted as transmission time interval (TTI). The total simulation horizon is set to be 2000 TTIs, where 1 TTI is set to be 10 ms. During each TTI, the positions of the client vehicles and buses are updated. We assume that the vehicles can have at most one active task. The air interface used in the simulation is 5G NR n78 with a 3500-MHz frequency band and a 20-MHz channel bandwidth. The dominant path model is used for estimating the SINR of the users at each TTI. Each user is assigned to the cell with the maximum SINR. Further details on the scheduling algorithms can be found in [49].

We consider two traffic scenarios, one with 50 client vehicles (i.e., off-peak scenario) while the other with 150 client vehicles (i.e., peak scenario). Three network latency requirements are set, namely, 50, 100, and 150 ms, and we measure the service rate for each case. During the simulation, the CFNs are co-located with 12 BSs, and the VFNs are carried by eight bus journeys. We compare the service rate among three deployment strategies.

1) *Deploying Strategy:* Using both CFNs and VFNs with standard capacity.
2) *Comparison Strategy—CFNs Only Standard (CP-COS):* Using CFNs only with standard capacity.
3) *Comparison Strategy—CFNs Only Extended (CP-COE):* Using CFNs only with 1.67 times of capacity.

Among the above strategies, *DS* corresponds to our deploying solution, while *CP-COS* and *CP-COE* are used for comparison. Different from the simulation setup in Section VII-B, here we control the capacity of single FNs as equal in strategy *DS* and *CP-COS*, while control the overall capacity of all the FNs as equal in strategy *DS* and *CP-COE* (i.e., $12 \times 1.67 \approx 12+8$). We compare the three deployment strategies in terms of service provision.

The service rates of different deployment strategies under various network and traffic scenarios are shown in Table IV. We average the presented results over 20 independent instances. The service rate is generally higher during the off-peak scenario than the peak scenario, with an average difference of 7%. This is because when the user number increases, the network resource will become more scarce. The increasing latency constraint causes an average decrease in service rate around 14% and 27% during the off-peak and peak scenarios, respectively, due to the increasing demand for computing resources.

Comparing different deployment strategies, *DS* has a higher service rate (up to 9.3%) than *CP-COS* and a slightly lower service rate (less than 6%) than *CP-COE*. Considering the relatively small performance difference between *DS* and *CP-COE*, it is possible to argue that the applicability of the deployment strategies depends on the economic feasibility. From the economic perspective, upgrading the cellular capacity in both ways means additional investment in the infrastructure. However, the mobility of VFC allows more flexible resource scheduling. Therefore, VFC can save long-term operational costs and thus, be more cost efficient than stationary deployment.

X. DISCUSSION

In this section, we discuss the demand uncertainty, the computational complexity of the capacity planning model, and the advantages and limitations of our work.

A. Uncertainty in the Demand

We estimated the demand for computing resources in Section V, with the assumption that the vehicular traffic would follow the same spatiotemporal distribution during the planning period. However, the vehicular traffic may change in the future, e.g., when the number of autonomous vehicles increases [50]. In this section, we detail how our framework would handle the uncertainty caused by seasonal changes and occasional events.

Seasonal Changes: The vehicular traffic and the demand for computing resources may vary with weather conditions, daylight hours, and holiday seasons (e.g., July in Finland). In Fig. 12, we use the vehicular traffic in Helsinki in January and February 2020 as examples and test whether the capacity plan made for January would be still suitable for February. In our experiment, we set the latency constraint to 100 ms and the relative operational cost to 1.0. We first compare the temporal traffic distribution between January and February. As shown in Fig. 12(a) and (b), there are two peak hours in January: one in the morning and another in the afternoon; however, in February, the amount of traffic during the peak hour in the afternoons is lower than in January. We then calculate the optimal capacity plans for January and February 2020,

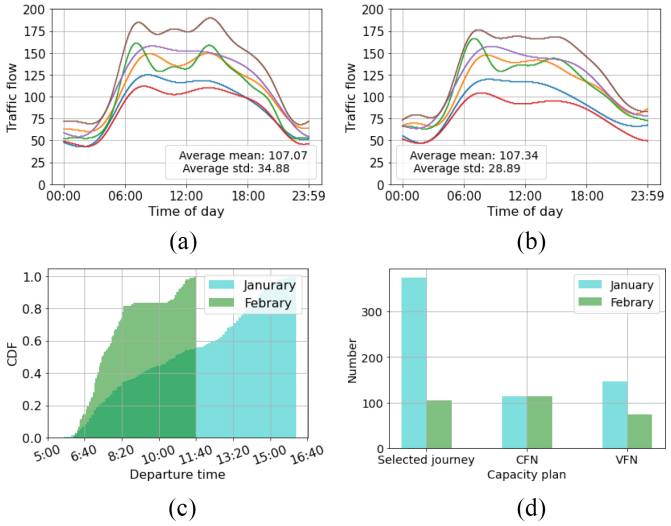

Fig. 12. Comparison of capacity plan for January and February 2020. (a) Traffic model in January 2020. (b) Traffic model in February 2020. (c) Comparison of departure time of selected bus journeys. (d) Comparison of journey number and FNs deployment.

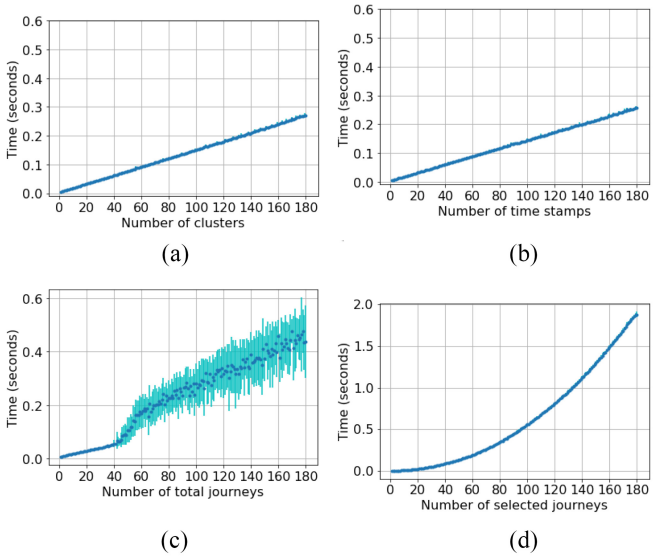

Fig. 13. Execution time versus the size of data, where blue points represent the mean values, and the cyan lines represent the variations. (a) Cost minimization execution time versus the number of clusters (with ten time steps and ten journeys). (b) Cost minimization execution time versus the number of time steps (with ten clusters and ten journeys). (c) Cost minimization execution time versus the number of bus journeys (with ten clusters and ten time steps). (d) Bus scheduling execution time versus the number of selected journeys.

and compare the schedules of the selected bus journeys in Fig. 12(c) and the numbers of FNs in Fig. 12(d). Fewer bus journeys and VFNs would be needed to satisfy the demand in February, especially for the afternoons. If the capacity plan made for January is applied in February, regardless of the change in the demand, it would generate a 2.05% higher operational cost than the optimal capacity plan based on February's traffic. If the change in the demand is significant, it is necessary to update the capacity plan. However, it depends on the operators to decide the threshold or a regular interval for updating the capacity plan.

Occasional Events: Apart from seasonal changes, occasional events can also cause uncertainty in the total demand. For example, when a football match is held, the city will be crowded with football fans, which would cause a temporary increase in the computing resource demand. Under these circumstances, it is not necessary to update the long-term capacity plan because of the occasional changes. Instead, it is more cost efficient to temporarily increase the supply in certain areas by deploying more VFNs. Compared with buses that have fixed routes and timetables, taxis and drones are more suitable in this case, since they can be easily routed to different places. We call it on-demand VFC and will leave it for future work.

B. Computational Complexity

The demand estimation module can be solved in polynomial time using the algorithm explained in Section V-C. The computational complexity of the cost minimization and the bus scheduling modules is evaluated for various data sizes. We use a commercially available computer equipped with an Intel Core i7-7700K CPU at 4.2-GHz frequency, where one thread out of eight is used during the measurements. We have repeated the measurements for 20 independent instances, and the presented results in Fig. 13 are averaged over these 20 instances. It can be seen that for the cost minimization module, the execution time increases linearly with the number of clusters and time steps. Furthermore, the execution time rises slightly faster than the linear relationship with the number of total journeys. For the bus scheduling model, the execution time increases quadratically with the number of selected journeys.

These linear and quadratic growth in complexity can be acceptable in real-world scenarios due to two reasons. First, the capacity planning is a long-term decision problem (i.e., in the order of months or years); therefore, there are no real-time constraints in the model. Second, even under stricter time constraints, it is possible to use high-power computing resources to decrease the execution time. In our experiments for the Helsinki downtown area, the execution time of demand estimation for each vehicular application combination is within 1 s. The average execution time of cost minimizing and bus scheduling is around 283 and 9 s, respectively, when we consider six clusters, 1440 time steps, and 5189 total journeys.

C. Advantages and Limitations of Our Work

Our capacity planning solution provides the following advantages. First, it minimizes the costs by complementing stationary FNs with mobile ones. Compared with the models that consider only stationary FNs [8], [16]–[19], [21], [22], our framework has the potential to reduce the long-term operational cost. Compared with the solutions that deploy FNs on all buses [6], [7], [25], [26], our solution selects a subset of buses to install FNs based on traffic information, which results in lower installation costs without loss in QoS. Second, by enabling the mobility of FNs, our solution provides

more flexibility in the capacity planning model, especially for handling the uncertainty in demand.

On the other hand, the limitations of our work mainly exist in two aspects. First, our regression-based model does not predict the demand in the future. Therefore, it is not possible to proactively update the capacity plan. As an extension of this work, we will use deep learning-based algorithms, such as the long short-term memory (LSTM) network [51] and graph convolutional network (GCN) [52], for predicting the future traffic flow and update the capacity plan accordingly. Second, our capacity planning method targets long-term FN deployment. In our future work, we will complement our current long-term deployment model with short-term on-demand scheduling (as detailed in Section X-A). We also believe that our data-driven approach in this article can be part of beyond 5G capacity planning, which leads to joint planning of computing and communication resources in the future.

XI. Conclusion

This work proposes a data-driven capacity planning framework that optimizes the deployment of stationary and mobile FNs. By exploiting the spatiotemporal fluctuations in demand, we minimize the installation and operational costs within the QoS requirements. We model the spatiotemporal distribution of vehicular traffic and the computing resource demand. We use a heuristic algorithm and ILP to find the cost-optimal solution. We validate our framework using real-world vehicular traffic data, vehicular applications, and bus timetables. Compared with the conventional solutions that rely on cellular FN deployments, the experimental results demonstrate the potential of our framework to reduce costs. The results also show that the deployment of mobile FNs saves operational costs at the expense of additional installation costs. Moreover, in the long term, more operational costs will be saved in the times and areas with higher traffic density and greater daily variation due to the dense deployment of VFNs.

References

[1] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global Internet Things Summit (GIoTS)*, 2017, pp. 1–6.

[2] L. Castiglione, P. Falcone, A. Petrillo, S. Romano, and S. Santini, "Cooperative intersection crossing over 5G," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 303–317, Feb. 2021.

[3] M. Atagoziyev, K. W. Schmidt, and E. G. Schmidt, "Lane change scheduling for autonomous vehicles," *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 61–66, 2016.

[4] M. Chiang, B. Balasubramanian, and F. Bonomi, *Fog for 5G and IoT*, 1st ed. Hoboken, NJ, USA: Wiley, 2017.

[5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[6] C. Zhu, G. Pastor, Y. Xiao, and A. Ylajaaski, "Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 58–63, Oct. 2018.

[7] C. Zhu *et al.* "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.

[8] M. Noreikis, Y. Xiao, and A. Ylä-Jaäiski, "QoS-oriented capacity planning for edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[9] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 6–9. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PERCOMW.2017.7917508

[10] Z. Zhang, Q. He, H. Tong, J. Gou, and X. Li, "Spatial-temporal traffic flow pattern identification and anomaly detection with dictionary-based compression theory in a large-scale urban network," *Transp. Res. C Emerg. Technol.*, vol. 71, pp. 284–302, Oct. 2016.

[11] H.-S. Zhang, Y. Zhang, Z.-H. Li, and D.-C. Hu, "Spatial-temporal traffic data analysis based on global data management using MAS," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 267–275, Dec. 2004.

[12] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3512–3524, Apr. 2019.

[13] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, p. 2423, Jul. 2018.

[14] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4942–4956, Oct. 2019.

[15] L. Mai, N.-N. Dao, and M. Park, "Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing," *Sensors*, vol. 18, no. 9, p. 2830, Aug. 2018.

[16] T. Chiu, W. Chung, A. Pang, Y. Yu, and P. Yen, "Ultra-low latency service provision in 5G fog-radio access networks," in *Proc. IEEE 27th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2016, pp. 1–6.

[17] D. Zhang, F. Haider, M. St-Hilaire, and C. Makaya, "Model and algorithms for the planning of fog computing networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3873–3884, Apr. 2019.

[18] F. Haider, D. Zhang, M. St-Hilaire, and C. Makaya, "On the planning and design problem of fog computing networks," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 724–736, Apr.–Jun. 2021.

[19] I. Stypsanelli, O. Brun, S. Medjiah, and B. J. Prabhu, "Capacity planning of fog computing infrastructures under probabilistic delay guarantees," in *Proc. IEEE Int. Conf. Fog Comput. (ICFC)*, 2019, pp. 185–194.

[20] M. Noreikis, Y. Xiao, and Y. Jiang, "Edge capacity planning for real time compute-intensive applications," in *Proc. IEEE Int. Conf. Fog Comput. (ICFC)*, 2019, pp. 175–184.

[21] M. M. Hussain, M. Alam, and M. M. Beg, "Vehicular fog computing-planning and design," *Procedia Comput. Sci.*, vol. 167, pp. 2570–2580, Jan. 2020.

[22] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago, "Efficient placement of edge computing devices for vehicular applications in smart cities," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, 2018, pp. 1–9.

[23] X. Hou *et al.*, "Reliable computation offloading for edge-computing-enabled software-defined IoV," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7097–7111, Aug. 2020.

[24] Z. Zhou, H. Liao, X. Wang, S. Mumtaz, and J. Rodriguez, "When vehicular fog computing meets autonomous driving: Computational resource management and task offloading," *IEEE Netw.*, vol. 34, no. 6, pp. 70–76, Nov./Dec. 2020.

[25] C. Zhu, Y.-H. Chiang, A. Mehrabi, Y. Xiao, A. Ylä-Jääski, and Y. Ji, "Chameleon: Latency and resolution aware task offloading for visual-based assisted driving," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9038–9048, Sep. 2019.

[26] C. Zhu, Y.-H. Chiang, Y. Xiao, and Y. Ji, "FlexSensing: A QoI and latency-aware task allocation scheme for vehicle-based visual crowdsourcing via deep *Q*-network," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7625–7637, May 2021.

[27] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[28] J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo, "Vehicular sensing networks in a smart city: Principles, technologies and applications," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 122–132, Feb. 2018.

[29] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70169–70184, 2019.

[30] R. Molina-Masegosa and J. Gozalvez, "LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications," *IEEE Veh. Technol. Mag.*, vol. 12, no. 4, pp. 30–39, Dec. 2017.

[31] *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol Specification, Version 14.2.2*, 3GPP Standard TS 36.331, Apr. 2017
[32] *System-Level Evaluations on Sidelink for NR V2X*, document 3GPP TSG RAN WG1 95, R1-1812210, Huawei, Spokane, WA, USA, Nov. 2018.
[33] L. Elefteriadou, *An Introduction to Traffic Flow Theory*, vol. 84. New York, NY, USA: Springer, Jan. 2014.
[34] "Docker." [Online]. Available: https://www.docker.com/ (accessed Aug. 31, 2021).
[35] S. Bunte and N. Kliewer, "An overview on vehicle scheduling models," *Public Transp.*, vol. 1, pp. 299–317, Nov. 2010.
[36] "Here." [Online]. Available: https://www.here.com/ (accessed Aug. 31, 2021).
[37] "Here Traffic API." [Online]. Available: https://developer.here.com/documentation/traffic (accessed Aug. 31, 2021).
[38] "TMS Data." [Online]. Available: https://vayla.fi/en/transport-network/data/open-data/road-network/tms-data (accessed Aug. 31, 2021).
[39] "High-Frequency Positioning." [Online]. Available: https://digitransit.fi/en/developers/apis/4-realtime-api/vehicle-positions/ (accessed Aug. 31, 2021).
[40] "Open Data—HSL.fi." [Online]. Available: https://www.hsl.fi/en/hsl/open-data (accessed Aug. 31, 2021).
[41] G. Jocher *et al.* "Ultralytics/yolov5: V3.1—Bug Fixes and Performance Improvements." Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370
[42] "Coco—Common Objects in Context." [Online]. Available: https://cocodataset.org/ (accessed Aug. 31, 2021).
[43] D. Gupta and R. J. Wala. "Image Segmentation Keras: Implementation of Segnet, FCN, UNet, PSPNet and Other Models in Keras." Dec. 2020. [Online]. Available: https://github.com/divamgupta/image-segmentation-keras
[44] "Cityscapes Dataset." [Online]. Available: https://www.cityscapes-dataset.com/ (accessed Aug. 31, 2021).
[45] "Opencv." [Online]. Available: https://opencv.org/ (accessed Aug. 31, 2021).
[46] "Handbrake." [Online]. Available: https://handbrake.fr/ (accessed Aug. 31, 2021).
[47] B. Alves. "Global Household Electricity Prices 2020." [Online]. Available: https://www.statista.com/statistics/263492/electricity-prices-in-selected-countries/ (accessed Aug. 31, 2021).
[48] (Gurobi Optim., Houston, TX, USA). *Gurobi Optimizer Reference Manual*, (2020). [Online]. Available: http://www.gurobi.com
[49] O. U. Akgul, W. Mao, B. Cho, and Y. Xiao. "A Data-driven Platform for Simulating Vehicular Fog Computing Environment." Jan. 2022. [Online]. Available: https://doi.org/10.36227/techrxiv.17829398.v1
[50] B. Friedrich, *The Effect of Autonomous Vehicles on Traffic*. Heidelberg, Germany: Springer, May 2016, pp. 317–334.
[51] X. Di, Y. Xiao, C. Zhu, Y. Deng, Q. Zhao, and W. Rao, "Traffic congestion prediction by spatiotemporal propagation patterns," in *Proc. 20th IEEE Int. Conf. Mobile Data Manag. (MDM)*, 2019, pp. 298–303.
[52] Q. Xie, T. Guo, Y. Chen, Y. Xiao, X. Wang, and B. Y. Zhao, "'How do urban incidents affect traffic speed?' A deep graph convolutional network for incident-driven traffic speed prediction," 2019, *arxiv:1912.01242*.

Wencan Mao received the B.E. degree in vehicle engineering from Wuhan University of Technology, Wuhan, China, in 2017, and the M.S. degree in mechanical engineering from Aalto University, Espoo, Finland, in 2019, where she is currently pursuing the Ph.D. degree with the Department of Communications and Networking.

Her current research interests include edge computing, Internet of Things, vehicular networking, resource allocation, and capacity planning.

Ozgur Umut Akgul received the B.S. degree in electronics engineering and electrical engineering and the M.S. degree in computer engineering from Istanbul Technical University, Sariyer/Istanbul, Turkey, in 2011 and 2014, respectively, and the Ph.D. degree in information technology from Politecnico di Milano, Milan, Italy, in 2019.

He is currently a Postdoctoral Researcher with the Department of Communications and Networking, Aalto University, Espoo, Finland. His research interests focus on optimization models, mathematical programming, and machine learning, with the application of these techniques to wireless network problems, such as wireless resource allocation, edge computing, anticipatory network optimization, infrastructure and resource sharing, and network slicing.

Abbas Mehrabi (Member, IEEE) received the B.Sc. degree in computer engineering from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2008, and the M.Sc. degree in computer engineering from Islamic Azad University (South Tehran Branch), Tehran, Iran, in 2010.

He is a Senior Lecturer with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. He was a Lecturer of Computer Science with Nottingham Trent University, Nottingham, U.K., from 2019 to 2020. From 2017 to 2019, he was a Postdoctoral Researcher with the Distributed and Mobile Systems Research Group in the Computer Science Department, Aalto University, Espoo, Finland. His research interests include mobile cloud/edge computing, Internet of Things, vehicular networking, and smart grid communications.

Dr. Mehrabi is a Fellow of U.K. Higher Education Academy.

Byungjin Cho received the Doctoral degree in communications engineering from Aalto University, Espoo, Finland, in 2016.

He is currently a Postdoctoral Researcher with the Department of Communications and Networking, Aalto University. His research interests include resource managements in networked systems using algorithmic decision theory.

Yu Xiao (Member, IEEE) received the Doctoral degree in computer science from Aalto University, Espoo, Finland, in 2012.

She is currently an Assistant Professor with the Department of Communications and Networking, Aalto University. Her current research interests include edge computing, wearable sensing, and extended reality.

Antti Ylä-Jääski received the Ph.D. degree from ETH Zürich, Zurich, Switzerland, in 1993.

From 1994 to 2009, he was with Nokia, Espoo, Finland, in several research and research management positions, with a focus on future Internet, mobile networks, applications, services, and service architectures. Since 2004, he has been a Tenured Professor with the Department of Computer Science, Aalto University, Espoo. His current research interests include mobile cloud computing, mobile multimedia systems, pervasive computing and communications, indoor positioning and navigation, energy-efficient communications and computing, and Internet of Things.