IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 30, 2021, accepted November 18, 2021, date of publication November 23, 2021,
date of current version December 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130125

A Real-Time Energy-Saving Mechanism in Internet of Vehicles Systems

LUCA CESARANO[1], ANDREA CROCE[1], LEANDRO DO CARMO MARTINS[2], DANIELE TARCHI[1], (Senior Member, IEEE), AND ANGEL A. JUAN[2]

[1]Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40136 Bologna, Italy
[2]IN3—Computer Science Department, Universitat Oberta de Catalunya, 08018 Barcelona, Spain

Corresponding author: Daniele Tarchi (daniele.tarchi@unibo.it)

The work of Luca Cesarano and Andrea Croce has been done during an abroad study period at Universitat Oberta de Catalunya, Spain, supported by Erasmus+ Study Programme of the European Union.

ABSTRACT Emerging technologies, such as self-driving cars and 5G communications, are raising new mobility and transportation possibilities in smart and sustainable cities, bringing to a new echo-system often referred to as Internet of Vehicles (IoV). In order to efficiently operate, an IoV system should take into account more stringent requirements with respect to traditional IoT systems, e.g., ultra-broadband connections, high-speed mobility, high-energy efficiency and requires efficient real-time algorithms. This paper proposes an energy and communication driven model for IoV scenarios, where roadside units (RSUs) need to be frequently assigned and re-assigned to the operating vehicles. The problem has been formulated as an Uncapacitated Facility Location Problem (UFLP) for jointly solving the RSU-to-vehicle allocation problem while managing the RSUs switch-on and -off processes. Differently from traditional UFLP approaches, based on static solutions, we propose here a fast-heuristic approach, based on a dynamic multi-period time scale mapping: the proposed algorithm is able to efficiently manage in real-time the RSUs, selecting at each period those to be activated and those to be switched off. The resulting methodology is tested against a set of benchmark instances, which allows us to illustrate its potential. Results, in terms of overall cost –mapping both energy consumption and transmission delays–, number of active RSUs, and convergence speed, are compared with static approaches, showing the effectiveness of the proposed dynamic solution. It is noticeable a gain of up to 11% in terms of overall cost with respect to the static approaches, with a moderate additional delay for finding the solution, around 0.8 s, while the overall number of RSUs to be switched on is sensibly reduced up to a fraction of 15% of the overall number of deployed RSUs, in the most convenient scenario.

INDEX TERMS Internet of vehicles, smart cities, real-time optimization, energy minimization, uncapacitated facility location problem.

I. INTRODUCTION

Each form of innovation aiming to achieve sustainability and to optimize the use of resources in responsible ways is defined as an eco-innovation form. The need for developing green-growth technological strategies in urban environments is raising day-by-day, and has the noble aim of improving citizens' quality of life, apart from ensuring that future generations will be able to meet basic and first-necessity needs –e.g., clean air, water, and biodiversity– thanks to adjustments like the reduction in traffic and in emission of greenhouse gases [1].

Modern technologies, like 5G and Internet of Things (IoT), can be employed to support the development of *green cities* [2]–[4]: a huge distributed cloud infrastructure made by thousands of smart devices (typically small and embedded ones), able to communicate with each other in real-time [5]. Among several application domains, the Internet of Vehicles (IoV) [6], [7], a sub-domain of IoT, can be described as a distributed and interconnected network that supports the need for sharing data created by vehicles –but could include also pedestrians, bikes, and other urban objects– in a real-time fashion with roadside units (RSUs). These are special facilities that act as edge computing nodes. They run different services and are able to jointly convey communications and process data, without requiring transfer to central cloud facilities [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

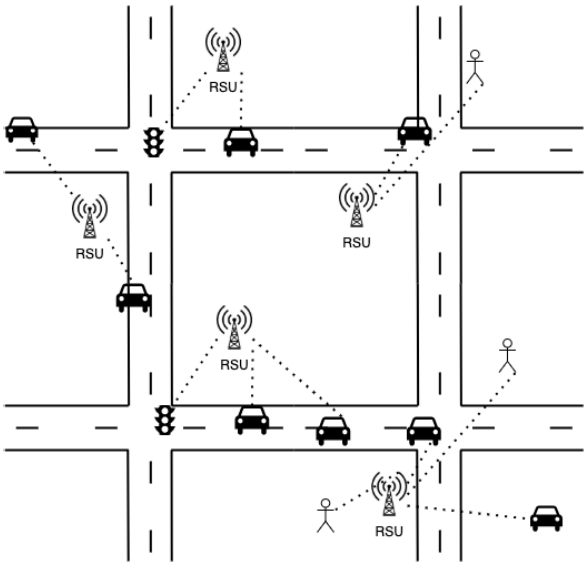


FIGURE 1. RSU management in the Internet of Vehicles scenario.

As shown in Figure 1, one of the main challenges that arise when solving an IoV-related problem is to efficiently manage the RSUs, so that an optimal service coverage of vehicles is provided with respect to some target Quality of Service (QoS) parameters –e.g., service coverage, throughput, low latency, or energy consumption [9]–[11]. In addition, roads are very dynamic environments: cities may be congested in some areas at specific times of a day, with vehicles moving at a different speeds. This dynamism of the environment must be taken into account as well.

The problem addressed in this paper aims at considering the IoV scenario by including a dynamic behavior in the problem definition. **The system is modeled as a series of consecutive periods, in which the RSUs configuration has to react and quickly adapt to the ever-changing traffic and connection requirements from the vehicles, while keeping the energy consumption low.** Thus, every few minutes, an efficient configuration of RSUs might need to be re-computed in a short amount of time, typically within a few seconds. In Figure 2, the same city blocks are depicted in three different successive periods, where vehicles and pedestrians are in movement and share the same space. Despite the facilities are fixed, the users' and vehicles' connection are updated over time –according to the dynamism of this environment– in order to minimize connection costs.

In particular, we aim at optimizing the RSUs activation through a proper switch-on and -off, enabling their energy consumption minimization, under the data-rate and mobility requirements of the vehicles. The system is modeled through a multi-period time-scale and, in the following, the problem to be solved is referred to as a Multi-Period Internet of Vehicle Problem (MPIoVP). To this aim, the MPIoVP is first introduced and modeled as a semi-dynamic Uncapacitated Facility Location Problem (UFLP). The UFLP is an *NP-hard* optimization problem in Operations Research, which covers a wide range of logistics and transportation applications. This classical problem was initially described in [12]. The goal of the UFLP is to find an optimal placement for a given set of facilities, dictated by an objective function and some constraints. Despite its original formulation has fixed inputs over time and does not have real-time constraints, the UFLP can be adapted for modeling the MPIoVP, by assuming that the inputs change over time. Hence, the MPIoVP becomes an optimization problem where we assume that the RSUs are first mapped on selected locations and then managed; the solutions need to be found frequently and quickly over time, giving the very dynamic nature of these environments. In addition to this we developed and proposed a suitable mobility model able to catch in a simple way the collective movement of vehicles within an urban area, by properly modeling vehicles with different mobility patterns.

In order to solve the MPIoVP, an extremely fast biased-randomized (BR) algorithm [13] is proposed. This solution method, which incorporates stochastic components, is then embedded into a multi-start framework. In this way, the resulting approach is able to periodically generate high-quality alternative solutions in *real-time*, i.e., in a few seconds. A series of computational experiments allow us to validate the efficiency of the proposed methodology.

The rest of the paper is organized as follows: Section II offers a review of some related works. In Section III, the first formal introduction of the MPIoVP is given. The biased-randomized algorithm, and the framework in which the algorithm is embedded into, is described in Section IV. Numerical results are presented and discussed in Section V, where both experimental setup and results are considered. Finally, Section VI presents some final considerations on our results and elaborates on how this research may be continued in the future.

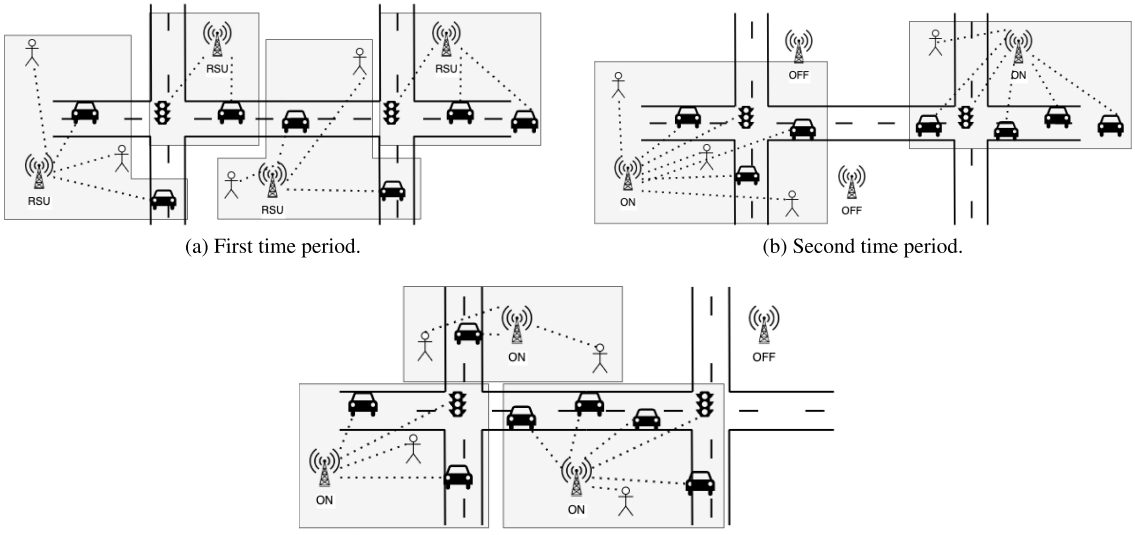
II. RELATED WORKS AND MAIN CONTRIBUTIONS

This section reviews previous publications in the context of Telecommunication Networks and IoV systems, with a particular focus on their optimization in dynamic scenarios. A brief background on BR algorithms is also given. It also discusses the main contribution of our work with respect to the literature.

A. TELECOMMUNICATION NETWORKS AND INTERNET OF VEHICLES

Dynamic scenarios in telecommunication systems require optimization algorithms capable of solving complex problems in short computing times. Telecommunication networks are one of the most important representative scenarios, where real-time resource allocation is of primary importance when implementing new services, leading to very challenging problems.

Among other models, used for modeling telecommunication problems, the Facility Location Problem (FLP) [14]



(a) First time period. (b) Second time period.

(c) Third time period.

FIGURE 2. The IoV scenario modeled as an UFLP, where users and vehicles are assigned to different RSUs.

has been extensively used in the past, showing its flexibility when considering realistic scenarios. As an example, the authors in [15] studied an FLP in the 5G cellular networks communication. They shown how to maximize the Wi-Fi coverage given a demand, and how to connect Wi-Fi equipment in order to ensure that the wireless connection is always available in an open city scenario. The authors in [16] aim at optimizing the resource utilization in a fog computing environment for respecting the tasks execution deadlines via assigning them to an optimal fog server using Software Defined Networking (SDN). Moreover, local and global load balance techniques are proposed. The simulation results revealed that the proposed system decreases the amount of tasks to be moved to the cloud and, as a result, there is a reduction in latency and in bandwidth consumption. In [17], the authors have modeled a video streaming system with QoS thresholds as a stochastic single-allocation p-hub median problem, by considering a simheuristic algorithm. A hub-and-spoke network is analyzed, in which numerous nodes exchange real-time multimedia data, and where the quantity of data sent from one node to another is a random variable.

Among several modern telecommunication systems, IoV systems represent one of the most challenging, due to their stringent time-scale and real-time service requirements. The IoV scenario has been introduced in [6], where the authors suggest IoV as a joint view of IoT and vehicular communications, a new system where the *vehicles* represent the *things* of the road environment. In [18], a hierarchical architecture for IoV scenarios is proposed, where the complexity of the scenario is highlighted through the design of different layers, each one managing a specific aspect. The interested reader could refer to [7], where the most important network architectures and applications for IoV are surveyed. It is indeed clear that IoV embraces several aspects. Among others we focus in this paper on the RSU operation [11], [19], which is usually faced in the literature through two main approaches. On the one side, the optimal RSU placement, involving the optimization of the RSUs position and their deployment in a given scenario [9], [20]–[26]. On the other side the optimal RSU selection, among a multitude, for the service implementation [27]–[32]. In particular our work focus on the second aspect aiming at investigating the possibility of optimally selecting the RSUs to be used by each vehicle while moving within the road scenario. In addition to this, an efficient energy saving mechanism is introduced aiming at switching off some of the RSUs while respecting the system constraints.

The RSU selection has been considered in [28], where the authors employ a simplified one-way road model to study the connectivity probability with a predefined set of RSUs placed along the way. In [32], the RSU selection is performed in a heterogeneous network scenario, where each vehicle has to select the best RSU for performing the computation offloading. The problem is solved through a multi-armed bandit algorithm, where the goal is to minimize the cell handover for avoiding service disruption. Often, the RSU selection is studied jointly with other key performance indexes. As an example, in [27] the authors consider the energy efficiency as a measure for optimizing the RSU selection, while in [31] the authors exploit full-duplex communications for optimally selecting the RSUs to be used by each vehicle. In [29], the authors proposed a Vehicle-to-Vehicle aided approach for optimizing the task offloading selection in an urban scenario where multiple RSUs are supposed to be scattered along the roads. Likewise, in [30] the authors extended the previous approach to consider mobile nodes acting as relay and/or processing nodes for implementing on-demand vehicular services.

Focusing instead on the energy efficiency aspects some works in the literature considered such a problem within

an IoV scenario. Among other works, in [8], the authors focus on the energy consumption control issues of MEC-enabled RSUs by introducing an energy-efficient scheduling framework for MEC-enabled IoVs to minimize the energy consumption of RSUs under task latency constraints. In [27], the authors addressed a low-complexity RSU and time slot assignment in vehicular networks with multiple RSUs. The goal is to minimize energy consumption across the RSUs. Another approach was considered in [33], where the authors provide an energy efficient offline downlink scheduling algorithm for store-carry-forward vehicles to satisfy the residual data requirement of the target vehicle moving in the uncovered area. A cluster-based approach is proposed to identify vehicles in energy favorable locations and multi-hop distance to relay vehicles. In [23], the authors propose a joint placement and sleep scheduling of RSUs in a VANET environment. In the paper the author formulate an optimization problem able to capture the behavior of a realistic scenario and propose an energy efficient candidate location selection algorithm to jointly perform placement and sleep scheduling of grid-connected solar powered RSUs. In [34], the authors considers the RSU ON/OFF scheduling problem in a sparse VANET. The proposed multi-level greed algorithm can significantly reduce the energy consumption of RSUs deployed and can provide a reference basis for energy efficient scheduling in VANETs.

One should notice that the optimal assignment of vehicles to RSUs in IoV scenarios has been barely studied. In [35], the authors studied an IoV scenario using UFLP modeling, but they did not considered the vehicular mobility or the impact of switching on and off the RSUs as the system evolves. As introduced in [36], RSUs alongside roads are used as wireless access points in an IoV network, providing communication services to vehicles inside its coverage area. Relating this problem to FLPs, the vehicles' demands for communication represent the inputs of our FLP. Since vehicles are in movement, these inputs are, henceforth, dynamic, and the outputs need to be periodically recomputed. Dealing with input uncertainties means dealing with real-world scenarios, which leads to the search for more robust or reliable solutions. In [37] the authors deal with uncertainties using random variables in its model: cost, demands, and distances. One of the few studies that addresses an optimization model to solve the RSU deployment problem is presented in [11]. A linear programming (LP) clustering algorithm, composed of three steps, was proposed. Their approach employs a utility function to measure and evaluate the total benefit from the RSU deployment. The stages are based on clustering, problem instance reduction –formulated as a single-node capacitated facility location problem–, and tasks assignment.

B. BIASED-RANDOMIZED ALGORITHMS

Biased-randomized algorithms constitute a relatively recent approach for fast generation of alternative solutions to complex optimization problems [38]. This methodology employs skewed probability distributions, such as the geometric and triangular ones, to extend the constructive behavior of deterministic heuristics [39]. Many applications of biased-randomized heuristics can be found in the literature [40]. The first use of a non-uniform probability distribution to induce a biased random effect into a constructive heuristic was given in [41], where the authors solved a capacitated vehicle routing problem. In their work, a savings-based heuristic was biased by introducing a geometric probability distribution during the constructive stage. From this study, BR heuristics have been proposed to solve a vast number of optimization problems in different applications, as well as to enhance some metaheuristic frameworks such as the ILS [42], the greedy randomized adaptive search procedure (GRASP) [43], [44], or the multi-start one [45], [46]. They have also been combined with learning mechanisms [47], and with parallel computing [48].

C. MAIN CONTRIBUTIONS

After discussing the emerging interest in IoV systems, their real-time requirements, and the need for developing fast optimization algorithms for the UFLP, **this paper proposes an original framework for real-time energy-efficient management of RSUs in a dynamic urban environment.** To this aim, the main achievements of our paper can be summarized as follows:

- A dynamic urban scenario has been modeled by considering the presence of multiple RSUs and vehicles. Each RSU is able to implement an energy saving mechanism, modeled as a two-state system. Differently from other papers in the literature, our model considers multiple geographical subareas, each one associated with a demand factor parameter that represents the amount of vehicles located in that subarea at a given time, as well as their service requests. This allows to create a model that can be easily manged when we have to select the RSUs to be switched off, and allowing to map the cells to each active RSUs.
- To the best of our knowledge, this is the first paper formulating the problem as an UFLP, where the minimization of both energy consumption and data-rate are considered in the cost function, through a proper management of the RSUs. The traditional UFLP approach has been extended towards a multi-period framework, characterized by a variable demand, by introducing a closing and an opening fee for modeling the cost to be paid for switching on and off each RSU.
- Based on a graph model, a random shift rule has been defined, allowing to properly map a realistic scenario where vehicles move preferably towards the downtown area at some time periods (e.g., in the morning). A suitable heatmap representation located in Rome (Italy) is also considered, helping to better understand the demands evolution.
- An original fast constructive savings-based heuristic is proposed. The heuristic has been also enriched with

biased-randomization techniques, and embedded into a multi-start framework.
- Finally, simulation results have been carried out, comparing five possible strategies, including both static and dynamic approaches.

III. SYSTEM MODEL AND PROBLEM FORMULATION

This section provides details on the model we have used to represent the IoV system.

A. SYSTEM MODEL

We focus on an urban scenario composed of several RSUs. These are distributed across the city, and identified through the set $\mathcal{R} = \{RSU_1, \ldots, RSU_i, \ldots, RSU_I\}$, whose positions (x_i^R, y_i^R) are fixed. We also consider a multitude of vehicles, identified through the set $\mathcal{V} = \{v_1, \ldots, v_m, \ldots, v_M\}$, whose positions $(x_m(t), y_m(t))$ are changing over time. While the RSUs can provide IoV services to the users (e.g., Internet access, edge processing, etc.), vehicles are likely to choose one of the RSUs within their coverage range to access to the requested service. The set of vehicles that can be connected to the ith RSU at time t is defined as:

$$\mathcal{V}_i^R(t) = \{v_m | d(v_m, RSU_i) \leq d^R\} \quad \forall v_m \in \mathcal{V}$$

where $d(v_m, RSU_i)$ is the distance between the ith RSU and any of the M vehicles, while d^R is the coverage range of the RSUs, which are assumed to be the same for all the RSUs and which depend on the communication technology (e.g., IEEE 802.11p, C-V2X [49]).

The distance between RSUs and vehicles drives the connection availability and the link quality by setting the expected data-rate. Given the Shannon capacity formula, the data-rate is a function, among other factors, of the distance, i.e.:

$$\rho_{im} = B_{im} \log_2 \left(1 + \frac{P_{tx}^i}{L(d_{im}) P_{N_m}}\right)$$

where B_{im} represents the bandwidth of the link between the ith RSU and the mth vehicle, P_{tx}^i is the transmission power of the ith RSU, $L(d_{im})$ is the path loss at a distance d_{im} between the ith RSU and the mth vehicle, and P_{N_m} is the noise power at the receiver side. Noise power can be defined as $P_{N_m} = N_T B_{im}$, where N_T is the thermal noise. A similar formula can be considered for the reciprocal link.

To reduce the impact of the energy consumption, we assume that each RSU can be switched on (active) or in stand-by state. When active, a RSU is able to service a certain number of vehicles. On the contrary, no vehicle can be serviced for a RSU in stand-by mode. Based on some previous works [50], [51], the power consumption of the i-th RSU can be modeled as follows:

$$P_i = \begin{cases} P_i^{sleep} & \text{if } RSU_i \text{ is in stand-by mode} \\ P_i^0 + \mu_i P_i^{active} & \text{if } RSU_i \text{ is switched on} \end{cases} \quad (1)$$

where P_i^0 is the basic circuit power consumption, μ_i is a proportionality coefficient dependent on the number of vehicles connected to the ith RSU and P_i^{active} is the power consumed by the ith RSU for managing the vehicles.[1] In the following, let us assume that:

$$P_i^{ON} = P_i^0 + \mu_i P_i^{active} - P_i^{sleep} \quad (2)$$

is the additional power consumed when the RSU is operative with respect to the stand-by mode. Additionally, a switching cost, corresponding to the power spent when switching from the stand-by to active mode, can be defined as [50]:

$$P_i^{sw} = E_i^{sw} \frac{N^{sw}}{\tau} \quad (3)$$

where E_i^{sw} is the energy spent during the switching process, and N^{sw} stands for the number of switches during a time period τ.

In order to reduce the transmission latency, each vehicle will always be assigned to the closest active RSU. Also, in order to minimize energy consumption, the algorithm decides which RSUs have to be activated or set in stand-by mode at each period. From the vehicle perspective, the selection of the RSU can be modeled through a proper cost function, aiming at considering the behavior of each link by focusing on distance. From the RSUs perspective, the operation of a RSU is driven by its energy consumption, while aiming at servicing all the vehicles in the area and switching on only the needed RSUs. Hence, while the distance parameter drives the link capacity, where a higher distance corresponds to a lower link capacity, the energy parameter refers to the amount of energy consumed in order to keep both the link and the RSU operative.

However, the RSU management problem cannot be modeled with a static assignment of RSUs and vehicles, due to the mobility of the latter. Thus, in order to take into account the dynamism of the system we divide the entire area $\mathcal{A}$ into J geographical subareas, named cells, where the jth cell is identified as $\mathcal{C}_j$, and $\mathcal{A} = \cup_j \mathcal{C}_j$. Within each cell, multiple vehicles are located at a given time t, where:

$$\mathcal{V}_j^C(t) = \{v_m | v_m \in \mathcal{C}_j\} \quad \forall v_m \in \mathcal{V}$$

is the set of vehicles within the jth cell at time t. In order to map the overall requests of the vehicles within a cell, we introduce a demand factor, d_j^t, which corresponds to the demand of the vehicles within the jth cell at time t. The demand can be considered as an overall factor mapping together the requested data-rate, the amount of services, and the connection quality. The higher the demand, the more resources should be reserved to the vehicle by a certain RSU. We consider to have a random demand in each cell, whose value is changing in time. Henceforth, the system must be re-optimized taking in consideration the vehicle mobility.

[1] P_i^{active} is a value that is supposed to include all the transmission, reception and processing power cost of a given RSU.

With this in mind, an assignment cost, based on the distance between a cell and a RSU, is taken into consideration, together with the demand of the cell itself. The total demand in a cell is the sum of all the individual demands of the vehicles currently located in that cell. Since the vehicles are continuously in motion, the demand in each cell varies over time. Considering an initial setting of demands, the goal is to determine which is the lowest-cost configuration of RSUs, i.e., which RSUs need to remain operative in order to minimize the sum of energy and delay costs.

B. PROBLEM FORMULATION

In the problem under study, we consider a dynamic evolution of the vehicular traffic over time. In particular, we are interested in taking into account changes in the vehicles' location, which might affect the aggregated demand associated with each cell.

We consider in the following that the time is organized in slots. Hence, $t \in \mathbb{N}$ represents the index of the time slot starting at $t \cdot \tau$, and having duration τ, while $\mathcal{T} = \{0, \ldots, t, \ldots, T-1\}$ is the whole time-span of the scenario under consideration. The problem is formulated so that the solving methodology is applied to multiple periods, i.e., to multiple slots having length τ.

Let us define the maximum amount of energy that can be consumed by the ith RSU as

$$r_i = P_i^{ON} \cdot \tau + E_i^{sw}$$

corresponding to the energy in the operational mode (with respect to the stand-by mode) in (2), plus the energy for switching on the RSU in (3). We can introduce a parameter, λ_i^t, defining a weight factor mapping the RSU_i state at time slot t, and depending on its mode (i.e., switched-on or in stand-by) in the previous time slot, i.e., $t-1$. If the ith RSU was in stand-by state at $t-1$, the cost at t is considered equal to $\lambda_i^t r_i$, with $\lambda_i^t = 1$. Otherwise, only a partial cost $\lambda_i^t r_i$ (i.e., a maintenance cost) is paid, where $0 < \lambda_i < 1$. By this definition of λ_i^t, we are able to map a higher cost for switching on the RSU rather than maintaining it operative. We can set a decision vector $\mathbf{Y}^t$, whose ith element is the variable y_i^t. This element is associated with the selection of a certain RSU_i at time slot t, defined as:

$$y_i^t = \begin{cases} 1 & RSU_i \text{ is switched on at time slot } t, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall RSU_i \in \mathcal{R}, \forall t \in \mathcal{T} \tag{4}$$

Then, it is possible to define the overall cost related to the activation of the RSUs in any time period:

$$\sum_{RSU_i \in \mathcal{R}} \lambda_i^t r_i y_i^t \quad \forall t \in \mathcal{T} \tag{5}$$

The cost in (5) is related to a fixed cost to be paid for either keeping operative or activating the RSUs. It has to be noticed that the number of switches is indirectly considered through the sum performed over all the RSUs. If not active, a RSU can be switched ON; hence, the overall number of switches corresponds to the sum of all the RSUs that are switched ON. We assume that the time slot is the minimum amount of time between two consecutive switches. Also, in order to take into account the load to be managed by each RSU –in terms of vehicles– an additional cost term should be introduced. It will consider the distance between the RSUs and the cells, thus modeling the link availability and quality between RSUs and vehicles. Notice that this value is fixed over time, while the vehicles and their related demand will be changing within each cell.

Let us consider a cost matrix $\mathbf{C}$, where a generic element c_{ij} stands for the cost between RSU_i and a cell j in terms of distance, so that $c_{ij} \propto d_{ij}$, where d_{ij} is the distance between the ith RSU and the jth cell (i.e., the higher the distance, the higher the cost).

Since each cell represents the geographical subarea in which the vehicles are positioned, we have to map the vehicle density and their communication needs for each cell. In order to do this, we introduce a demand vector, identified as $\mathbf{D}$, which indicates the aggregated demand per cell.[2] To be more specific, in order to consider the dynamic behavior of the system we identify with $\mathbf{D}^{(t)}$ the demand vector at time slot t. The elements of this vector, d_j^t, represent the demand of a cell j at time slot t. The higher the number of vehicles in that cell, the higher the associated demand. The total demand of a cell j within the time slot t is the sum of the individual demands of the vehicles in that cell, i.e.:

$$d_j^t = \sum_{v_m \in \mathcal{C}_j} \tilde{d}_m^t \quad \forall \mathcal{C}_j \in \mathcal{A}, \ \forall t \in \mathcal{T}$$

where $\tilde{d}_m^t$ is the individual demand of the mth vehicle at time slot t.

Let us introduce now a decision matrix $\mathbf{X}^t$, where the (i,j)-th element is the variable x_{ij}^t standing for the connection between the ith RSU and the jth cell in the time slot t:

$$x_{ij}^t = \begin{cases} 1 & \text{if } RSU_i \text{ is connected to cell } j \text{ in time slot } t, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall RSU_i \in \mathcal{R}, \quad \forall \mathcal{C}_j \in \mathcal{A}, \ \forall t \in \mathcal{T} \tag{6}$$

This corresponds to the possibility of connecting all vehicles in the set $\mathcal{V}_j^C(t)$ (i.e., belonging to the jth cell at time slot t) with the ith RSU. Therefore, for a given time slot $t \in \mathcal{T}$, we can define our problem as:

$$\left[\mathbf{X}^{t*}, \mathbf{Y}^{t*}\right] = \arg\min_{\mathbf{X}^t, \mathbf{Y}^t} \left\{ \sum_{RSU_i \in \mathcal{R}} \lambda_i^t r_i y_i^t + \sum_{RSU_i \in \mathcal{R}} \sum_{\mathcal{C}_j \in \mathcal{A}} c_{ij} d_j^t x_{ij}^t \right\} \tag{7}$$

[2] In the context of our paper, the demand corresponds to the generic resources requested by the vehicles in that cell. Such resources could represent bandwidth, computing, or any other resource required for setting up the service.

subject to the following constraints:

$$\mathbf{C1}: \sum_{\mathcal{C}_j \in \mathcal{A}} x_{ij}^t = 1 \quad \forall RSU_i \in \mathcal{R} \tag{8}$$

$$\mathbf{C2}: x_{ij}^t \leq y_i^t \quad \forall RSU_i \in \mathcal{R} \tag{9}$$

$$\mathbf{C3}: x_{ij}^t, y_i^t \in \{0,1\} \quad \forall RSU_i \in \mathcal{R}, \quad \forall \mathcal{C}_j \in \mathcal{A} \tag{10}$$

$$\mathbf{C4}: r_i, c_{ij}, d_j^t \in \mathbb{N} \quad \forall RSU_i \in \mathcal{R}, \quad \forall \mathcal{C}_j \in \mathcal{A} \tag{11}$$

where:

- $\left[\mathbf{X}^{t*}, \mathbf{Y}^{t*}\right]$ correspond to the values minimizing the cost function in (7) for having minimized both RSU activation and cell to RSU allocation. As previously introduced, the cost term c_{ij} is used for modeling the cost between a group of vehicles in a given $\mathcal{C}_j$ and the RSU_i. The cost can be considered as proportional to the distance (i.e., considering the Shannon law as a reference, higher is the distance, lower is the data-rate, hence, higher is the cost for that link). Since the cost is associated with the connection between one cell and one RSU we have to multiply it with the number of vehicles, i.e., the demand, in that cell for taking into account the potential numerosity of vehicles in that cell.
- **C1** in (8) ensures that, inside each period, a cell j can be associated with only one RSU i;
- **C2** in (9) guarantees that, if there is a link between a cell j and an RSU i, then the RSU is operative;
- **C3** in (10) indicates that the two decision variables are binary
- **C4** in (11) stands that r_i, c_{ij}, d_j^t are integer variables.

The minimization in (7) is performed at each time slot. The aggregated effect contributes to the minimization of the whole time scale on which it is evaluated.

C. DEMANDS FORMULATION

The higher the number of vehicles in a cell, the higher the weight of a connection between that cell and the closest RSU will be. We model such condition through the demand $\mathbf{D}$, representing the amount of connections required by all vehicles in that cell. However, the urban environment is dynamic. Hence, the demand varies over time, introducing the necessity of a fast heuristic algorithm that can easily adapt the RSUs configuration over time. The second member of the objective function in (7) takes into account the demands vector $\mathbf{D}$ associated with the cells of the system. At the beginning of every time interval, the demand is supposed to change with a probabilistic pattern, modeling the overall vehicles behavior within an urban area. More details on how this occurs are provided next.

1) THE MOBILITY MODEL

With the purpose of modeling the movement of vehicles inside a city, we considered a graph representation of the area, $G = \{E, V\}$, where the vertexes represent the cells, and the edges represents the demand shift among adjacent cells (i.e., the mobility model of the vehicles). We also took into account the following hypotheses:

- the demand vector has the same cardinality as the number of cells in the topology;
- demands can shift only among adjacent cells, including the perimeter of the region being considered;
- the total demand is constant over time, meaning that when part of the demand leaves the system (i.e., crosses the perimeter), an identical quantity of demand enters from the opposite border, i.e.:

$$\sum_{\mathcal{C}_j \in \mathcal{A}} d_j^t = \|\mathbf{D}\|_1 \quad \forall t \in \mathcal{T}$$

where $\|\mathbf{D}\|_1$ corresponds to the sum of all the demands in all the cells, assumed to be constant during the whole time scale.[3]

In particular we assume that starting from a certain number of vehicles at time slot t only a portion of them can move to one of the adjacent cells. Therefore, the model assumes that some of the road users can be fixed or slowly moving, i.e., parked cars, vulnerable road users. The vehicle selected at the previous step can potentially move to another cell; among them only a portion really moves, modeling the fact that for some reasons a vehicle can decide to stop. Finally, among those vehicles selected to move a portion is supposed to go towards the usual destination (e.g., downtown) while the remaining go back for other reasons. We think this model, based on three probabilistic parameters, allows us to efficiently map the behavior in an urban scenario.

Although our methodology can be applied in any other scenario, in the computational experiments we will assume the existence of a high-density area, namely *downtown area*, which attracts vehicles (i.e., demands) in certain periods of the day (e.g., business hours) and from which vehicles departure in other periods, e.g., once most business close. Let us define the cell identifying the downtown area as $\mathcal{C}_{DT}$, having coordinates (x^{DT}, y^{DT}).

Algorithm 1 presents the mobility model. Given a certain number of demands, i.e., vehicles, at time slot t, (line 1), by focusing on the jth cell (line 3) we assume that only a portion of them, $\xi_j^t \in [0,1]$, can move to one of the adjacent cell (line 4). Hence, the vehicles that can potentially move towards adjacent cells from the jth cell at time t correspond to $\lceil \xi_j^t \cdot d_j^t \rceil$, where the ceiling is considered in order to have always an integer number of potential moving vehicles different from 0. Therefore, the model assumes that some of the road users can be fixed or slowly moving, i.e., parked cars, vulnerable road users, while others *can* move to a different cell. The selected $\lceil \xi_j^t \cdot d_j^t \rceil$ demands are assumed to move with a given probability, that we set to be equal to $\chi_j^t \in (0,1]$ (line 5). In other words, we assume that $\lceil \xi_j^t \cdot d_j^t \rceil$ vehicles move with probability χ_j^t, while they remain in the same cell with probability $(1 - \chi_j^t)$. As explained before, in our computational experiments we

[3]This corresponds to say that in the whole area we have a certain number of vehicles that move within it. While the overall number of vehicles is constant in the whole area, they can move from one cell to another.

assume the existence of an attracting cell, named *downtown area* (line 2). At certain hours of the day, vehicles will move towards that area. In order to properly model this feature, we assume that α_j^t is the probability that the selected vehicles move *towards* the downtown cell when in jth cell at time t, while $(1 - \alpha_j^t)$ is the probability that they move *in the opposite direction* (line 6).[4] Based on two random values (lines 7-8), the demands for each cell are shifted following the shifting probability and the probability to move towards the destination area. In particular, based on the comparison between a random variable and the probability to move towards the downtown area, one of the two functions SHIFTTOWARD() and SHIFTFARTHER() are executed (lines 9-17).

Algorithm 1 Pseudo-Algorithm of the Mobility Model

1: $\mathbf{D}^{(t)}$ is a list of demands at time t
2: $\mathcal{C}_{DT}$ is the destination cell
3: $\mathcal{C}_j$ is the j-th cell
4: $\xi_j^t \in [0, 1]$ is the quantity of demand that shifts across adjacent cells
5: $\chi_j^t \in (0, 1]$ is the probability that the quantity of demand shifts
6: $\alpha_j^t \in (0, 1]$ is the probability that vehicles move toward the destination area
7: *randShift* is a random number between (0, 1]
8: *randAlpha* is a random number between (0, 1]
9: **for all** $d_j^t \in \mathbf{D}^{(t)}$ **do**
10: **if** *randShift* $\leq \chi_j^t$ **then**
11: **if** *randAlpha* $\leq \alpha_j^t$ **then**
12: SHIFTTOWARD$(d_j^t, \mathcal{C}_j, \mathcal{C}_{DT}, \xi_j^t)$
13: **else**
14: SHIFTFARTHER$(d_j^t, \mathcal{C}_j, \mathcal{C}_{DT}, \xi_j^t)$
15: **end if**
16: **end if**
17: **end for**

In Algorithms 2 and 3 the SHIFTTOWARD() and SHIFTFARTHER() functions are detailed. We assume that $\mathcal{C}_\psi$ is a cell located closer to the destination area than $\mathcal{C}_j$, while $\mathcal{C}_\omega$ is a more distant cell. Hence, in SHIFTTOWARD() the demands at $t + 1$ are properly updated by increasing the quantity of demands of the closer-to-the-destination cell, and accordingly decreasing for the originating cell $\mathcal{C}_j$. Similarly, when SHIFTFARTHER() is executed the demands are increased at $t + 1$ for the more distant cell(s), and decreased for the originating cell.

Algorithm 2 Pseudocode of the SHIFTTOWARD Function

1: **function** SHIFTTOWARD$(d_j^t, \mathcal{C}_j, \mathcal{C}_{DT}, \xi_j^t)$
2: $d_\psi^{t+1} += \lceil \xi_j^t \cdot d_j^t \rceil$
3: $d_j^{t+1} -= \lceil \xi_j^t \cdot d_j^t \rceil$
4: **end function**

Algorithm 3 Pseudocode of the SHIFTFARTHER Function

1: **function** SHIFTFARTHER$(d_j^t, \mathcal{C}_j, \mathcal{C}_{DT}, \xi_j^t)$
2: $d_\omega^{t+1} += \lceil \xi_j^t \cdot d_j^t \rceil$
3: $d_j^{t+1} -= \lceil \xi_j^t \cdot d_j^t \rceil$
4: **end function**

[4] As a remark, we assume that each cell has only one possible cell allowing to reduce the distance towards the downtown area. Instead multiple cells allowing to increase the distance from the downtown area can exist; in this case the demands are equally shared among those cells.

TABLE 1. Parameters setting for the numerical example of demands shifting.

Parameters	Values
$\lvert\mathcal{T}\rvert$	20
cells	16
$\lVert\mathbf{D}\rVert_1$	80
(x^{DT}, y^{DT})	$(2, 2)$
ξ_j^t	0.4
χ_j^t	0.5
α_j^t	0.75

2) A NUMERICAL EXAMPLE

In order to verify the effectiveness of the mobility model, we have carried out a numerical experiment using the parameters in Table 1, where the cells are organized as a 4×4 square. For simplicity, we assume that the parameters remain fixed throughout the whole numerical example for all the cells, while the demand changes following the mobility model previously discussed.

Let us assume that the initial demands in period $t = 0$ are given by the matrix:

$$\Delta^{(0)} = \begin{bmatrix} 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 \\ 5 & 5 & \mathbf{5} & 5 \\ 5 & 5 & 5 & 5 \end{bmatrix}$$

that is a more effective representation of the demands in the geographical area, organized in a matrix corresponding to their geographical mapping, in which the bold element represents the downtown area.[5] Based on this initial configuration and the aforementioned parameters setting in Table 1, a potential shifting during the next period is:

$$\Delta^{(1)} = \begin{bmatrix} 3 & 3 & 5 & 4 \\ 3 & 3 & 5 & 8 \\ 10 & 7 & \mathbf{5} & 5 \\ 5 & 4 & 5 & 5 \end{bmatrix}$$

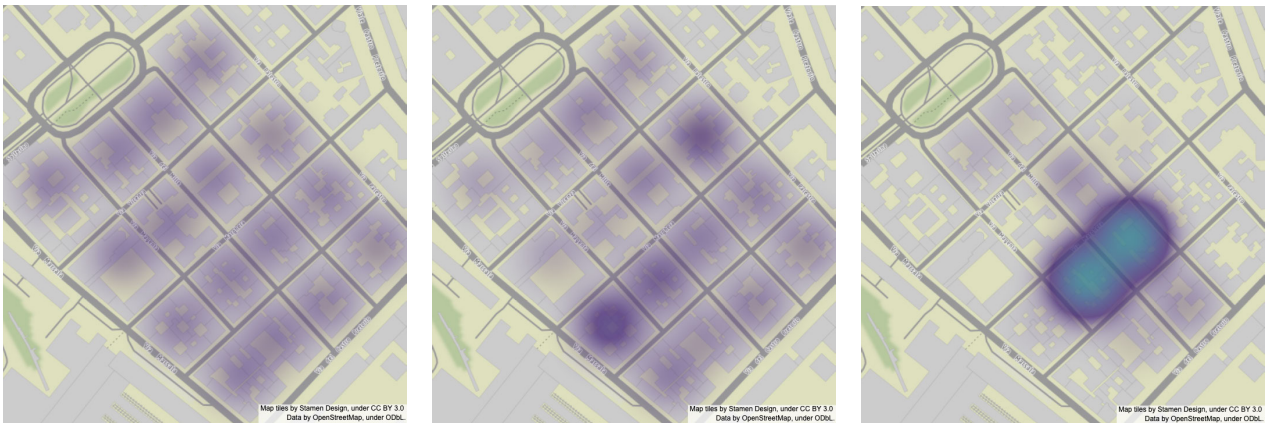
Similarly, after successive shifting processes, a potential configuration at $t = 19$ is:

$$\Delta^{(19)} = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 4 & 1 \\ 1 & 31 & \mathbf{33} & 0 \\ 0 & 2 & 4 & 0 \end{bmatrix}$$

Notice that the sum of all demands remains constant throughout the whole time span, while the downtown area, placed at (x^{DT}, y^{DT}), ends up being the most populated.

To better understand the concept shown in a matrix form, Figure 3 displays a geographical heatmap of a neighborhood belonging to the city of Rome (Italy). The heatmap in Figure 3a represents the initial scenario, where all the demands are well distributed all over the considered area. As time evolves, the demands shift towards the downtown area.

[5] We can hypothesize that demands in $\mathbf{D}$ and Δ are mapped so that $\Delta_{i,j} = \mathbf{D}_{i+4\cdot j}$



(a) Period $t = 0$. (b) Period $t = 1$. (c) Last period $t = 19$.

FIGURE 3. Heatmaps of traffic density at different time periods.

IV. SOLVING METHODOLOGY

The problem addressed in this paper considers a time variability of the input demands, thus leading to a dynamic and challenging problem. Since the UFLP is *NP-hard*, the use of exact methods is not an efficient strategy to solve large-scale instances in short computing times. Moreover, our scenario requires a real-time approach for a practical implementation. Therefore, we propose a fast constructive heuristic to solve the UFLP applied to the dynamic scenario previously identified. This heuristic is latter enriched with biased-randomization techniques, and embedded into a multi-start framework [52].

A. CONSTRUCTIVE HEURISTIC

The heuristic we propose is based on the concept of switching off facilities according to a *balance* factor, called *savings*, which is used to guide the search for feasible solutions. The procedure is shown in Algorithm 4. The procedure considers the configuration of RSUs in the previous time slot, as well the demand for the new time slot, where $sol_t = [\mathbf{X}^t, \mathbf{Y}^t]$ is the solution of the cost minimization problem in (7), i.e., the switched-on RSUs and the assignments of cells to RSUs. At the beginning of a time slot, the heuristic starts from a *virtual* scenario in which all the facilities are switched-on, which is hypothetical and highly expensive (line 2). The idea behind starting from this *all-active* configuration relies on the fact that switching off RSUs is computationally less expensive than activating them –this is because when a RSU is switched off, only the cells previously assigned to it have to be re-allocated to another RSU. The cost of this scenario is then calculated and used later as a reference (line 3). First, the cost of each facility is updated based on their state in the previous time slot (i.e., if it was switched on, in stand-by, or remains in the on state, lines 4-11). The *savings* are then updated based on the COMPUTESAVINGSLIST() function (line 12), which is detailed in Algorithm 5. Based on the new savings value, the solution is updated. The process is repeated until all RSUs in the list have been checked (lines 13-19). The list of savings is sorted in descending order, and the RSU with the highest positive savings is selected to be switched off. Once a RSU is switched off, the affected cells are re-assigned to alternative RSUs, which implies a re-computation of the individual savings given the new allocation scenario. Once re-calculated, the savings list is re-sorted, and this process is repeated until the end of the sorted list. Considering that the sort function adopted in our algorithm is $O(n\log(n))$, being n the number of elements in the savings list, we can assume that the worst-case bound for the running time is $O(n) + O(n\log(n)) = O(n\log(n))$.

Algorithm 4 Proposed Constructive Saving-Based Heuristic

1: **function** GENERATENEWSOL($sol_{t-1}, \mathbf{D}^t$)
2: $newSol \leftarrow allOpenSol$ ▷ sol with all facilities open
3: $cost(newSol) \leftarrow$ COMPUTECOST($newSol, \mathbf{D}^t$)
4: **for all** $y_i^t \neq 0$ **do**
5: **if** $y_i^{t-1} == 1$ **then**
6: $\lambda_i^t \leftarrow (0, 1)$
▷ Any value lower than 1. In the following, $\lambda_i^t = 0.5\ \forall\mathcal{R}$
7: **else**
8: $\lambda_i^t \leftarrow 1$
9: **end if**
10: Set $\lambda_i^t \cdot r_i$
11: **end for**
12: $savingsList \leftarrow$ COMPUTESAVINGSLIST($sol_{t-1}, \mathbf{D}^t, newSol, \{\lambda_i^t \cdot r_i\}_{\forall\mathcal{R}}$)
13: **while** $savingsList \neq \emptyset \wedge \sum_i y_i^t > 1$ **do**
14: $savingsList \leftarrow$ SORT($savingsList$)
▷ Sort from highest to lowest saving
15: $nextFacility \leftarrow savingsList[0]$
16: $newSol \leftarrow$ close $nextFacility \in newSol \wedge$ reallocate $\mathcal{V}^R_{nextFacility}(t)$
17: $cost(newSol) \leftarrow$ COMPUTECOST($newSol, \mathbf{D}^t$)
▷ The cost of $newSol$ is updated by considering the savings when closing $nextFacility$
18: $savingsList \leftarrow$ COMPUTESAVINGSLIST($sol_{t-1}, \mathbf{D}^t, newSol, \{\lambda_i^t \cdot r_i\}_{\forall\mathcal{R}}$)
19: **end while**
20: **return** $newSol$
21: **end function**

In Algorithm 5 the selection of the operative RSU is described by creating a *savingsList*. In case a RSU is switched-on (line 3), its savings is computed (line 4). If this savings is positive (line 5) the RSU is added to the *savingsList*

(line 6). Since negative savings values imply a higher cost in allocation settings, they are discarded.

Algorithm 5 Implementation of COMPUTESAVINGSLIST Function.

1: **function** COMPUTESAVINGSLIST(sol_{t-1}, $\mathbf{D}^t$, $newSol$, $\{\lambda_i^t \cdot r_i\}_{\forall \mathcal{R}}$)
2: $savingsList \leftarrow \emptyset$
3: **for all** $y_i^t \neq 0$ **do**
4: $savings_i \leftarrow$ COMPUTESAVINGS(sol_{t-1}, d_j^t, $\lambda_i^t \cdot r_i$)
5: **if** $savings_i > 0$ **then**
6: $savingsList \leftarrow$ ADD(RSU_i) ▷ list of facilities with positive savings
7: **end if**
8: **end for**
9: **return** $savingsList$
10: **end function**

The savings computation is performed for every operative RSU, as detailed in Algorithm 6: the cost of switching on / maintaining a RSU, plus the cost of assigning vehicles multiplied by their demand, minus their real-location cost to alternative facilities multiplied by their demand.

Algorithm 6 Implementation of COMPUTESAVINGS Function

1: **function** computeSavings(sol_{t-1}, d_j^t, $\lambda_i^t \cdot r_i$)
2: $allocationCost_i \leftarrow$ Sum of the demand weighted allocation cost for all the nodes assigned to RSU_i
3: $reallocationCost_i \leftarrow$ Sum of the demand weighted allocation cost of when nodes assigned to RSU_i are assigned to their best-alternative RSU
4: $savings_i \leftarrow \lambda_i^t \cdot r_i + allocationCost_i - reallocationCost_i$
5: **return** $savings_i$
6: **end function**

The described heuristic is sequentially applied in each single period. Notice that the final status at each period (demands and RSU configuration) constitutes the inputs of the new optimization problem in the subsequent period. The heuristic algorithm considers as the most expensive scenario the one in which demands are scattered homogeneously across the whole topology: in this situation, the algorithm activates a large number of RSUs, which increases the total operation cost. On the contrary, a situation in which vehicles' demands are concentrated around a specific area allows the algorithm to put in stand-by status many RSUs in remote areas with a low demand. This way, the algorithm is able to automatically adjust the RSU configuration to the actual requirements in each period.

B. BIASED-RANDOMIZATION

The proposed heuristic is a deterministic procedure which will always switch off the RSU with the largest savings value. In order to incorporate a random behavior to guide the search and, hence, provide a balanced exploration of the solution space, a biased-randomized heuristic has been implemented [13]. By extending the deterministic heuristic into a probabilistic algorithm through biased-randomization techniques, the logic behind the original heuristic is respected [45]. In our case, a geometric probability distribution is employed to smooth the selection of elements from the savings list. Consequently, the RSUs to be switched off are selected from the savings list according to a specific probability.

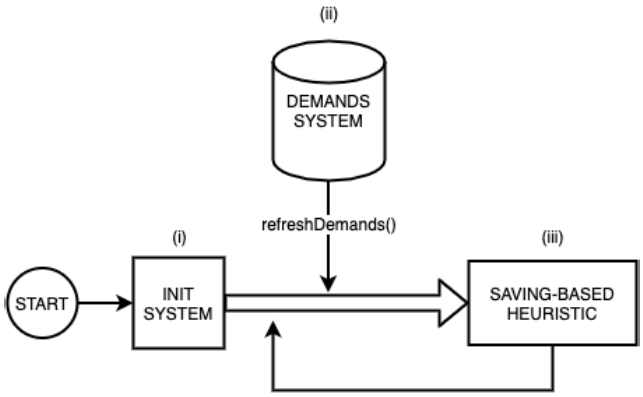


FIGURE 4. General architecture of our approach.

In the case of the geometric distribution, a single parameter $\beta \in (0, 1)$ is considered. When β approaches to 1, the behavior is similar to the greedy one of the deterministic heuristic. On the contrary, when β takes values close to 0, the behavior is similar to that of a uniform randomization process. The notable results can be obtained for intermediate values of this parameter. Given this particularity, different solutions are generated as more iterations of the biased-randomized algorithms are executed.

To exploit this particularity, this heuristic is embedded into a multi-start framework. Such a strategy relies on the generation of multiple solutions until a stop criterion is met. As a result, the best-found solution is returned at the end of this iterative process.

Figure 4 depicts the high-level flow of our approach. The main steps are: *(i)* initialize the system (only in the first period); *(ii)* consider the current value of the vehicles' demands; and *(iii)* apply the proposed algorithm to generate a new solution for the current period. Steps *(ii)* and *(iii)* are repeated in each time period. Accordingly, Algorithm 7 presents the proposed algorithm, where the demands are previously computed for each period using $\mathcal{T}$ (line 2). Then, the solutions are generated according to the current demands (lines 6-8). In Algorithm 8, the implementation of the objective cost function is reported, to be used multiple times within the Algorithm 7.

One should notice that, in a real-life scenario, demands may be refreshed at each time interval. This can be achieved in different ways, e.g., *(i)* by exploiting an external database that is periodically queried; *(ii)* by using RSU sensing capabilities to estimate the demands in real-time; and *(iii)* by employing a predictive model that forecasts the new values. Herein, we assume that demands change following the mobility model introduced in Section III-C1.

C. BENCHMARKING APPROACHES

Apart from the proposed solution methodology, other static approaches have been also considered as benchmarks to test

Algorithm 7 Fast Heuristic for Multi-Period Implementation of the RSU Activation Problem

Input: $r_i, d_j^t, c_{ij}, sol_0(\forall RSU_i \in \mathcal{R}, \forall \mathcal{C}_j \in \mathcal{A})$
Output: *total_cost*
1: $total\_cost \leftarrow 0$
2: **for** $t \leftarrow \{0, 1, 2, \ldots, T-1\}$ **do**
3: **for all** $\mathcal{C}_j \in \mathcal{A}$ **do**
4: $d_j^{t+1} \leftarrow$ REFRESHDEMANDS(d_j^t)
5: **end for**
6: $cost(sol_t) \leftarrow$ COMPUTECOST(sol_t, d^{t+1})
7: $sol_{t+1} \leftarrow$ GENERATENEWSOL(sol_t, d_j^{t+1})
8: $cost(sol_{t+1}) \leftarrow$ COMPUTECOST(sol_{t+1}, d_j^{t+1})
9: **if** $cost(sol_{t+1}) \geq cost(sol_t)$ **then**
10: $sol_{t+1} \leftarrow sol_t$
11: **end if**
12: $total\_cost \leftarrow total\_cost + cost(sol_{t+1})$
13: **end for**

Algorithm 8 Pseudocode of the COMPUTECOST function

1: **function** COMPUTECOST(sol_t, d_j^t)
2: $cost \leftarrow \sum_{RSU_i \in \mathcal{R}} \lambda_i r_i y_i^t + \sum_{RSU_i \in \mathcal{R}} \sum_{\mathcal{C}_j \in \mathcal{A}} c_{ij} d_j^t x_{ij}^t$
3: **return** *cost*
4: **end function**

the effectiveness of our algorithm. To this aim, the following five approaches have been defined:

1) *The single-open static approach*: in this approach, a single RSU, usually one located at the downtown area, is always operative.
2) *The all-open static approach*: in this approach, every RSU is switched-on at any time period. Hence, cells will always be assigned to their closest RSU (which will be always on). Of course, this approach offers a high cost, since all the RSUs are operative in every period.
3) *The keep-the-first static approach*: in this approach, the system will be optimized only once, in the first period. Thereafter, the first configuration of RSUs is kept for the rest of the periods, while the cost of this fixed configuration in each period is updated according to the new demands and the cost of keeping operative the selected RSUs. In principle, this approach will be less costly than the previous one. However, it does not adapt to variations in the vehicles' demands.
4) *The partially-dynamic approach*: as an enhancement of the previous approach, in this one, we re-optimize the system configuration after a series of k consecutive periods, with $k > 1$. In order to apply this approach in practice, it becomes necessary to provide a good balance between computational time and the quality of the recommended configuration.
5) *The fully-dynamic approach*: this approach is similar to the previous one, while $k = 1$. For achieving this, a multi-start strategy is needed in order to find a near-optimal configuration in each period and in real time. We can expect this approach to provide better results than the previous ones, although it requires a considerable effort in terms of quickly re-optimizing the system after periods that might span just a few minutes apart from each other.

V. NUMERICAL RESULTS

This section describes a series of computational experiments that, together with the analysis of the associated results, show the efficiency of the proposed approach.

A. EXPERIMENTAL SETUP

With the goal of testing the proposed approach, we have used the MED benchmark instances, which were originally introduced for the p-median problem [53]. The MED benchmark instances have been also employed in the UFLP context [54], being them largely studied thenceforth. Each instance is labeled with two numerical values, where the first identifies the amount of possible node locations, while the second is related to the cost for switching on each RSU. Despite not directly related with the IoV scenario, their usage has several advantages. First of all, MED instances are widely used in the UFLP evaluation, allowing us to compare our approach with the best-known / optimal values for the UFLP. Moreover, due to their generic definition, they can be used for testing different application scenarios including the previously described ones.

In the following, each instance will be identified as *n-sw*, where n is the number of location in that instance and *sw* is the switching-on cost scheme. Mapped to our problem, each location represents a potential RSU to be switched on. According to [53], we assume that the number of possible RSUs are 500, 1000, 1500, 2000, 2500, or 3000. We also assume that the switching cost scheme *sw* is equal to 10, 100, or 1000. Notice that the higher the switching-on cost scheme, the cheaper the switching-on of a RSU. Indeed, the switching cost of a generic ith RSU, can be related to the instances values n and *sw* through the rule [53]:

$$r_i = \frac{\sqrt{n}}{sw} \times 10000,$$

where *sw* represents the switching-on cost scheme, and n is the number of RSUs in the system. For example, the instance 500-10 considers that each RSU has a switching cost

$$r_i = \frac{\sqrt{500}}{10} \times 10000 \approx 22361.$$

Since r_i depends on both the opening cost and the switching cost, we assume that r_i, as derived from the values n and *sw*, represents the cost expressed in milliJoule. In the case of the previous example, and considering that τ is equal to 10 seconds, we have that both energy terms are of the order of 10 J. In the following, we have considered that the scenario is composed by n RSUs, that the demand for each potential RSU location d_j^t is equal to 50 at $t = 0$, and that this demand is supposed to change following the mobility model introduced in Section III-C1. For the computational experiments, we have set $\xi_j^t = 0.4$, $\chi_j^t = 0.5$, and $\alpha_j^t = 0.75$, as indicated in Table 1. In addition to this, the λ_i^t value used

TABLE 2. Parameters setting for the experimental results.

Parameters	Values
$\lvert\mathcal{T}\rvert$	20
# RSUs	{500, 1000, 1500, 2000, 2500, 3000}
sw	{10, 100, 1000}
τ	10 s
d_j^0	50
ξ_j^t	0.4
χ_j^t	0.5
α_j^t	0.75
λ_i^t	0.5

for evaluating the maintenance cost in (5) has been set to 0.5 $\forall t \in \mathcal{T}$. For the sake of clarity, the parameters used in the following experimental results, and previously introduced, are summarized in Table 2.

B. EXPERIMENTAL RESULTS

The computational environment employed for evaluating our approaches is an Intel Core i7-8550U processor with 16 GB of RAM. The goal of our experiment is twofold. On the one hand we want to verify the effectiveness of the proposed saving-based heuristic in terms of computational speed. This is of primary importance since the solution method has to work in a highly dynamic scenario, hence a real-time algorithm is necessary. On the other hand we want to verify the performance of the proposed solution in terms of cost reduction. Moreover, since the cost is composed of two terms, one modeling the data-rate and the other the energy consumption, both are verified separately for understanding their impact on the considered solution.

The first set of computational experiments relies on measuring the performance of our deterministic approaches when incorporating the demands shifting model into the problem. As introduced in Section IV, the savings-based procedure has been tested in different benchmarking approaches. Since the *single-operative* and *all-operative* static approaches consider a single operative RSU and all the RSUs operative over time, respectively, they do not include the savings-based heuristic as a guide.

Table 3 shows the computational results for the five different approaches. At this stage, the planning horizon is set to 20 periods, i.e., $T = 20$. For each test instance and approach, we can see: the average objective function value (OF) of all periods, the average number of operative RSUs ($|\mathcal{R}|$) across all periods, and the average computational time required to generate the results, in seconds. The average OF cost were divided by $1e6$ in order to provide easier-to-read results. The *gap* columns present the comparative performance of the different scenarios according to their respective average OF values.

In Table 3, we consider the *keep-the-first* approach as the *naive* approach against which other scenarios are compared. In this approach, the system is optimized at the beginning of the planning horizon, and the resulting configuration of operative RSUs and assigned vehicles is kept for the remaining periods. Due to the high variability of the switching-on cost, given by each scheme, it is convenient to analyze the results accordingly.

When activating RSUs is an expensive operation (i.e., instances with the suffix -10), the *all-on* approach has a higher cost than the *keep-the-first* approach –see column *gap (3)-(2)*, where it is noticeable that the cost is increased up to 142.3% for the all-on approach. By activating all the RSUs in the system, the assignment cost becomes null, since each node is assigned to itself. However, there are still demands to be supplied, which generates additional variable costs into the system. Since the activation cost is too large in this scenario, the higher the difference between the number of operative RSUs, the higher the gap among the respective instances will be (e.g., in instance $3000-10$ the activated RSUs are 329 out of 3000, i.e., around 11% of the total). By analyzing the performance of both *dynamic* approaches against the *keep-the-first* approach, it is clear that both of them outperform this naive approach, being the *fully-dynamic* approach the one offering a better overall performance. This behavior is expected, since the system is re-optimized in each period. It is worth to be noticed that the *fully-dynamic* approach with respect to the static *keep-the-first* approach allows a gain up to 11% in terms of overall cost, while the active RSUs remain of a similar amount. It has to be noticed that despite the number of activated RSUs is the same in average, in case of dynamic approaches they are selected on a period-by-period basis.

Another extreme scenario is the one in which the activation cost of RSUs is cheap (i.e., instances with the suffix -1000). In this case, there are no significant differences among the approaches, since the number of open RSUs exceeds 95% of all facilities in all cases. Hence, the main differences among the cost of these approaches rely on how the assignment of nodes is set –specially those which are not activated as RSUs. Particularly, the *keep-the-first* performs slightly better than the *all-on* approach, since a fraction of the fixed cost can be saved for not opening a few RSUs –and, then, the assignment of those RSUs that are not activated does not imply expensive variable costs; in this case a moderate gain around 3% can be achieved. Regarding the dynamic approaches, they present similar performance, since the majority of their RSUs are operative, which generates minimal differences in their variable costs.

The last scenario refers to the case in which the activation cost is moderate (i.e., the instances with the suffix -100). Unlike previous scenarios, the *all-on* approach presents better performance, on average, than the *keep-the-first* approach. Since the cost of activating a RSU is not too expensive, a large number of RSUs is frequently deployed. However, there are some cases in which the difference between the number of potential RSUs to be activated and the number of already operative RSUs is reasonably large (e.g., instances $2000-$, $2500-$, and $3000-100$). In these cases, the cost is mainly generated by the many assignments, implying higher variable costs. Therefore, for this cost scheme, the magnitude

TABLE 3. Computational results in terms of cost, facilities and completion time for the different approaches.

instance	single-on (1) avg.			all-on (2) avg.			keep-the-first (3) avg.			partially-dynamic (4) avg.			fully-dynamic (5) avg.			*gap*				
	OF	$\lvert\mathcal{R}\rvert$	time	OF	$\lvert\mathcal{R}\rvert$	time	OF	$\lvert\mathcal{R}\rvert$	time	OF	$\lvert\mathcal{R}\rvert$	time	OF	$\lvert\mathcal{R}\rvert$	time	(3)-(1)	(3)-(2)	(3)-(4)	(3)-(5)	(4)-(5)
500-10	148.2	1	0.0	5.9	500	0.0	5.9	206	0.0	5.3	241	0.0	5.2	245	0.1	2,430.1%	0.6%	-9.8%	-11.0%	-1.3%
1000-10	287.9	1	0.0	16.7	1,000	0.0	12.5	239	0.0	11.6	288	0.1	11.5	293	0.2	2,205.1%	33.3%	-7.0%	-7.9%	-0.9%
1500-10	343.6	1	0.0	30.6	1,500	0.0	18.9	259	0.0	17.9	310	0.2	17.7	315	0.5	1,717.7%	61.7%	-5.4%	-6.2%	-0.8%
2000-10	509.3	1	0.0	47.1	2,000	0.0	24.6	283	0.0	23.6	328	0.4	23.5	335	0.9	1,970.8%	91.3%	-4.0%	-4.6%	-0.6%
2500-10	510.3	1	0.0	65.8	2,500	0.0	29.9	307	0.0	28.6	362	0.7	28.5	366	1.3	1,607.0%	119.9%	-4.2%	-4.6%	-0.5%
3000-10	594.1	1	0.0	86.4	3,000	0.0	35.7	329	0.1	34.6	373	0.8	34.4	381	1.7	1,566.1%	142.3%	-3.0%	-3.4%	-0.5%
500-100	148.2	1	0.0	0.6	500	0.0	0.6	496	0.0	0.6	491	0.0	0.6	491	0.0	24,300.1%	0.8%	0.2%	0.1%	-0.1%
1000-100	287.9	1	0.0	1.7	1,000	0.0	1.7	944	0.0	1.7	929	0.1	1.7	928	0.2	16,384.0%	-2.1%	-1.7%	-2.2%	-0.4%
1500-100	343.6	1	0.0	3.1	1,500	0.0	3.2	1,299	0.0	3.1	1,305	0.2	3.1	1,306	0.5	10,626.9%	-2.4%	-2.7%	-3.4%	-0.7%
2000-100	509.3	1	0.0	4.8	2,000	0.0	4.9	1,628	0.0	4.7	1,599	0.4	4.7	1,597	0.9	10,313.3%	-1.9%	-3.1%	-3.8%	-0.8%
2500-100	510.3	1	0.0	6.7	2,500	0.0	7.1	1,724	0.0	6.6	1,786	0.7	6.5	1,788	1.4	7,126.9%	-5.3%	-7.2%	-8.1%	-1.1%
3000-100	594.1	1	0.0	8.8	3,000	0.0	9.0	1,981	0.0	8.5	1,997	1.0	8.4	1,999	2.3	6,509.7%	-2.4%	-5.4%	-6.3%	-1.0%
500-1000	148.2	1	0.0	0.1	500	0.0	0.1	496	0.0	0.1	496	0.0	0.1	496	0.0	177,711.6%	0.6%	0.0%	0.0%	0.0%
1000-1000	287.9	1	0.0	0.2	1,000	0.0	0.2	993	0.0	0.2	993	0.1	0.2	993	0.2	133,984.2%	0.5%	0.0%	0.0%	0.0%
1500-1000	343.6	1	0.0	0.4	1,500	0.0	0.4	1,468	0.0	0.4	1,468	0.2	0.4	1,468	0.4	91,947.2%	1.7%	0.0%	0.0%	0.0%
2000-1000	509.3	1	0.0	0.6	2,000	0.0	0.6	1,953	0.0	0.6	1,952	0.4	0.6	1,952	0.8	91,111.6%	2.0%	0.0%	0.0%	0.0%
2500-1000	510.3	1	0.0	0.8	2,500	0.0	0.8	2,414	0.0	0.8	2,413	0.7	0.8	2,413	1.3	67,159.5%	3.0%	0.0%	0.0%	0.0%
3000-1000	594.1	1	0.0	1.0	3,000	0.0	1.0	2,899	0.0	1.0	2,896	0.8	1.0	2,896	1.8	60,274.5%	3.0%	0.0%	0.0%	0.0%
Average	-	-	0.0	-	-	0.0	-	-	0.0	-	-	0.4	-	-	0.8	39,385.9%	24.8%	-2.9%	-3.4%	-0.5%

of both the activation and assignment costs results in better performance for the *all-on* approach against the *keep-the-first*. Nevertheless, when analyzing the performance of the dynamic approaches, both of them present a better average performance than the remaining approaches, showing a gain of 8.1% in the 2500-100 scenario.

Now, considering the complete benchmark instances –from the different activation schemes– we can assert, from column *gap (3)-(1)* to *gap (3)-(5)*, that the *fully-dynamic* approach shows overall the best performance. Since this approach re-optimizes the solution in every period, it is expected to be more efficient. However, it is noticeable how its efficiency worsens as the activation cost of RSUs decreases. Regarding computational times, the time required by the *fully-dynamic* approach is still low, being only 0.8 seconds slower than the *keep-the-first* approach. When comparing our two dynamic approaches, in column *gap (4)-(5)*, the *fully-dynamic* approach outperforms the *partially-dynamic* in 0.5%, on average. However, it requires 0.4 additional seconds to obtain these results. It is also to be noticed that the results are achieved by reducing the number of switched-on RSUs, that in some cases are just a fraction of around 15% of the overall number of RSUs deployed in the system.

Figure 5 presents, for all solving approaches, a box-plot of the corresponding results. The *single-on* approach is not included due to its numerical scale being too high. As we can notice, the *all-on* approach is clearly inefficient, mainly when the activation cost of RSUs is expensive. Moreover, we can notice that both the *fully* and *partially-dynamic* approaches are able to generate better results than the *keep-the-first*. Therefore, for this set of benchmark instances, the *fully-dynamic* is considered the best approach to cope with the MPIoVP, since it is able to generate better results on the average.

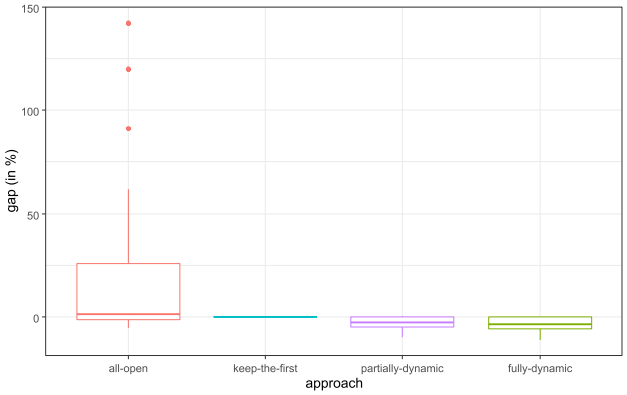
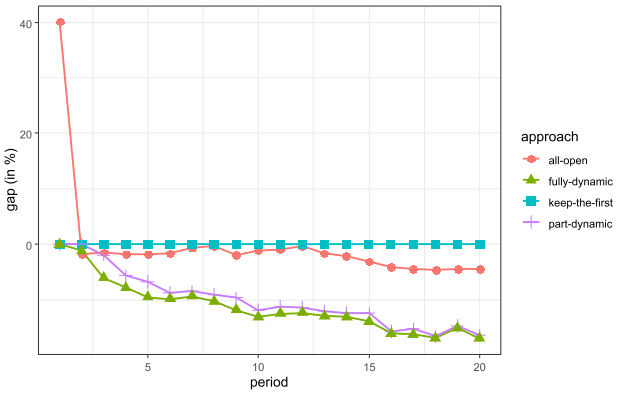


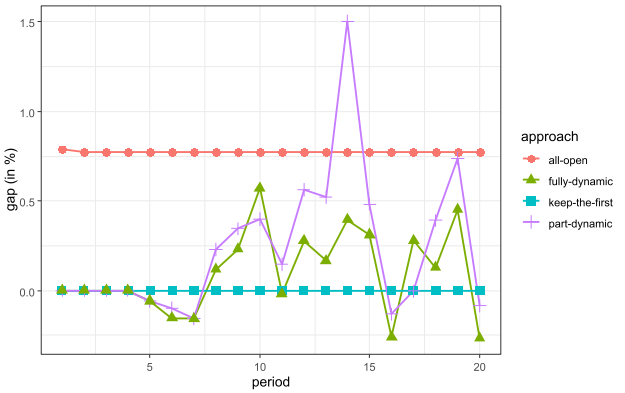
FIGURE 5. The variability of the average results for each scenario, in terms of gap.

For the problem instance composed of 500 RSUs, Figure 6 presents the convergence of the OF value over the periods. Similarly to what we did in Table 3, the comparisons are performed by considering the *keep-the-first* as the reference approach.

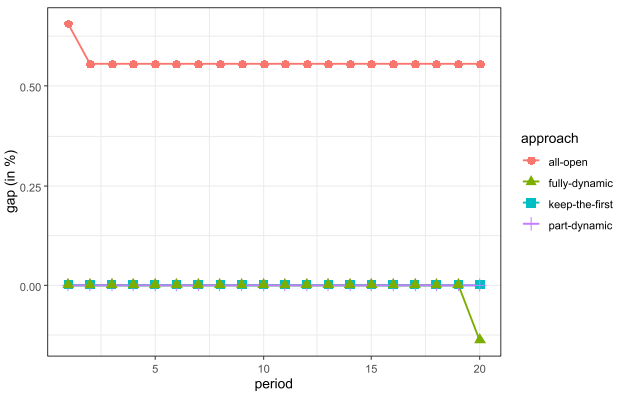
As we can see in Figure 6a –which represents the results obtained for the expensive activation scenario–, the *fully-dynamic* approach does not only provide better overall performance, but it is also able to generate better configurations for all the periods over time, achieving up to 17% of improvement in periods 18 and 20. Similarly, when comparing the *partially-dynamic* with the *keep-the-first* approaches, the same behavior is noticed, with up to 16% of improvement in periods 18 and 20. These conclusions support the efficiency of the dynamic strategies when demands vary over time. In addition, by analyzing Figure 6c one can conclude that, for the cheapest activation scenario, the performance of some approaches does not present any significant difference, since almost all the RSUs are deployed



(a) Instance *500-10* (expensive activation cost).

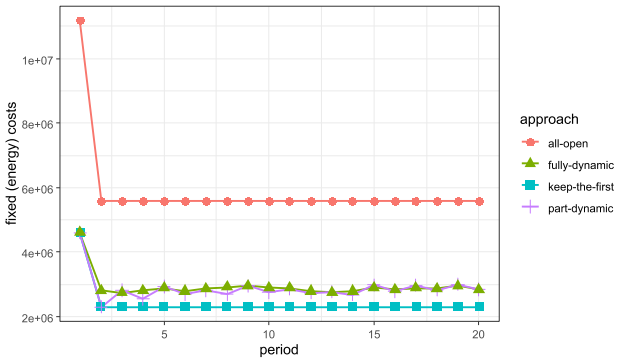


(b) Instance *500-100* (moderate opening cost).

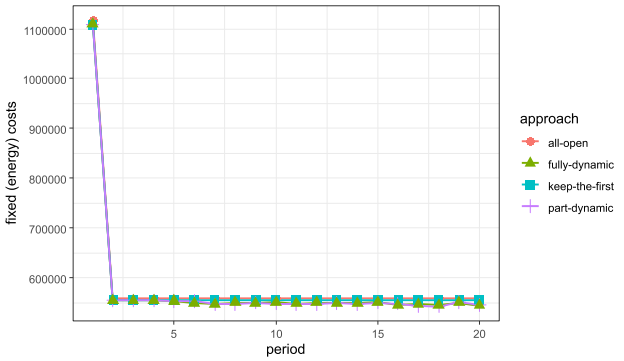


(c) Instance *500-1000* (cheap opening cost).

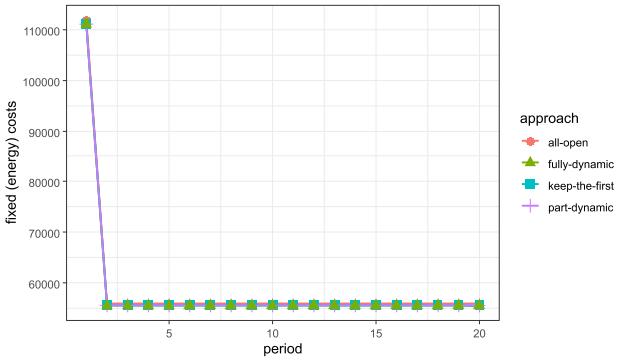
FIGURE 6. The convergence of the OF value over the periods, in terms of gap, for problem instance 500.



(a) Instance *500-10* (expensive activation cost).



(b) Instance *500-100* (moderate opening cost).



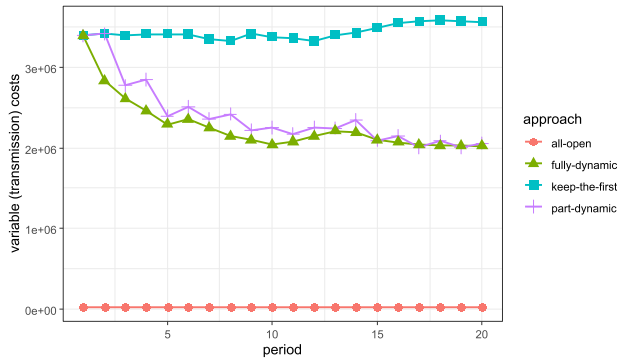
(c) Instance *500-1000* (cheap opening cost).

FIGURE 7. The energy consumption (fixed costs) over time, for problem instance 500.

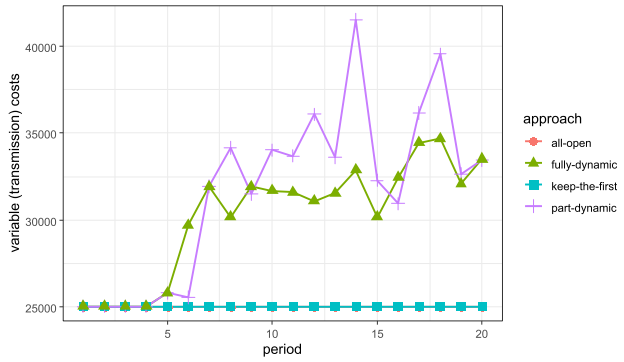
in any case. Notice, however, that the *fully-dynamic* approach presents a better performance again, mainly in the last period of the planning horizon. Finally, when analyzing the results under moderate activation cost (Figure 6b), one can observe that the *dynamic* strategies are able to properly react to the system changes, according to the demands shift.

In Figures 7 and 8, the solution costs over the periods are depicted in terms of energy consumption and transmission costs, respectively. As for the previous figures, the *single-on* approach is not included due to its high numerical scale. In our case, the energy consumption relates to the fixed costs, since it refers to the activation and deactivation of RSUs. Similarly, the transmission costs refer to the variable costs, representing the connection of vehicles to active RSUs.

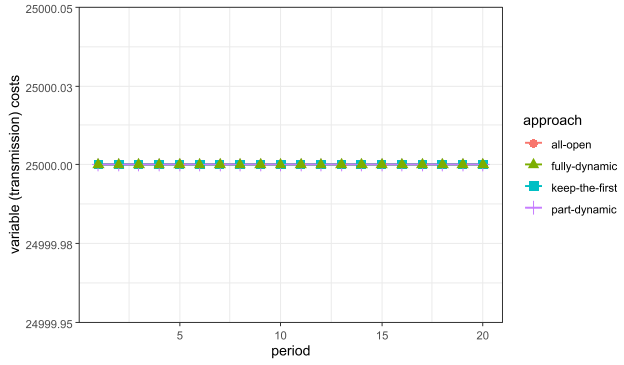
The following analysis aims at measuring the efficiency of the proposed approaches in terms of energy consumption costs. For the first scenario (Figure 7a), when the activation of RSUs has a higher cost, the *all-open* approach is certainly the less efficient approach –since all the RSUs are kept activated over time. Besides, it is noticed that its fixed cost is reduced from the first period on, since only the cost for keeping them active is incurred –rather than the activation cost– when they are previously activated. The same reduction cost is noticed for the remaining strategies. Regarding the *dynamic* approaches, both of them present a similar pattern. In terms of fixed cost, the *keep-the-first* presents the best performance in this scenario. However, the transmission costs incurred



(a) Instance *500-10* (expensive activation cost).



(b) Instance *500-100* (moderate opening cost).



(c) Instance *500-1000* (cheap opening cost).

FIGURE 8. The transmission (variable) costs over time, for problem instance 500.

by this configuration result in a less efficient approach, considering the complete panorama. For the remaining opening cost schemes (Figures 7b and 7b), the methodologies result into very similar fixed costs solutions. However, the *dynamic* approaches offer a better performance, in general. This is particularly the case for the moderate opening schema (Figure 7b).

Regarding the transmission costs, in the case of the expensive activation cost of RSUs (Figure 8a), one can notice how the dynamic approaches are capable to adapt to changes in demands. Despite being the cheaper approach in transmission costs –since all the RSUs are constantly activated–, the *all-open* approach is the more expensive approach in fixed costs. The *keep-the-first* approach presents, in the first periods, a performance which is similar to the one associated with the dynamic strategies. However, its transmission costs are continuously degraded, since this *static* approach does not adapt to the system changes by adjusting the decisions on the placement of RSUs made during the first period –when the demands are well distributed over the city. On the other hand, the transmission costs for both *partially-* and *fully-dynamic* approaches present similar convergence behavior, showing the *fully-dynamic* a better performance than the *partially-dynamic* over time, in general.

In the case of the moderate activation cost of RSUs (Figure 8b), the *all-open* and *keep-the-first* approaches present the best performance. However, the costs for all strategies are of similar magnitude. Also, when considering energy consumption costs under the same scenario, the dynamic strategies overcome the other strategies. In this way, the fixed counterpart of the solution costs pays-off the variable ones. Finally, when the opening cost is cheap (Figure 8c), transmission costs are approximately the same for all approaches. This is due to the fact that, for each approach, the majority of available RSUs are activated.

By analyzing both the fixed and variable costs separately, it is noticeable how conflicting they are, especially for the cheapest activation cost scenario, where the proposed approaches present the opposite behavior. In this case, for instance, while the *all-open* is the best approach regarding the variable costs –since all the RSUs are activated and, therefore, transmission costs are low–, the same approach requires the largest amount of fixed costs. Hence, the selection of the best approach is based on the analysis provided for Figure 6.

VI. CONCLUSION AND FUTURE RESEARCH

In the context of smart cities, this paper has analyzed the RSU activation problem in IoV scenarios when a dynamic behavior is considered. The goal was to optimize the configuration of the RSUs in each period, as a response to the time-evolving vehicles' communication needs. The underlying optimization problem has been modeled as a rich version of the UFLP, in which both RSUs activation (i.e., energy consumption) costs and transmission costs (between the RSUs and the vehicles) are considered. Since this is an *NP-hard* and large-scale problem for which new solutions need to be re-computed in real-time, the paper proposes the use of a fast optimization algorithm. This algorithm makes use of an efficient heuristic, which is then extended to a biased-randomized algorithm and, finally, embedded into a multi-start framework. This allows us to generate multiple high-quality solutions in just a few seconds. A series of numerical experiments are carried out to illustrate the benefits of the proposed algorithm. Our approach, which is dynamic, outperforms other more static approaches. The obtained results show how noticeable reductions in cost can be obtained by adopting a biased-randomized multi-start methodology as the one presented here. The proposed algorithm is fast and easily parallelizable. In addition, it only requires the fine-tuning of one single parameter.

Future works include extending the proposed solution method into an agile optimization strategy. The idea behind agile optimization techniques refers to the parallelization of biased-randomized algorithms. When embedded into a parallel framework, several solutions –with different characteristics– can be generated in the same wall-clock time as the one required by the original algorithm. Therefore, the best-found solution is returned.

REFERENCES

[1] (2020). *A Smart and Sustainable World Through ICT*. IEEE Sustainable ICT White Paper. [Online]. Available: https://sustainableict.ieee.org

[2] Q. Wu, G. Y. Li, W. Chen, D. W. K. Ng, and R. Schober, "An overview of sustainable green 5G networks," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 72–80, Aug. 2017.

[3] X. Liu and N. Ansari, "Toward green IoT: Energy solutions and key challenges," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 104–110, Mar. 2019.

[4] A. Srivastava, M. S. Gupta, and G. Kaur, "Green smart cities," in *Green and Smart Technologies for Smart Cities*, 1st ed., P. Tomar and G. Kaur, Eds. Boca Raton, FL, USA: CRC Press, 2019, pp. 1–18.

[5] Z. Ning, X. Kong, F. Xia, W. Hou, and X. Wang, "Green and sustainable cloud of things: Enabling collaborative edge computing," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 72–78, Jan. 2019.

[6] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Seoul, South Korea, Mar. 2014, pp. 241–246.

[7] B. Ji, X. Zhang, S. Mumtaz, C. Han, C. Li, H. Wen, and D. Wang, "Survey on the internet of vehicles: Network architectures and applications," *IEEE Commun. Standards Mag.*, vol. 4, no. 1, pp. 34–41, Mar. 2020.

[8] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling," *IEEE Netw.*, vol. 33, no. 5, pp. 198–205, Sep./Oct. 2019.

[9] T. J. Wu, W. Liao, and C. J. Chang, "A cost-effective strategy for road-side unit placement in vehicular networks," *IEEE Trans. Commun.*, vol. 60, no. 8, pp. 2295–2303, Aug. 2012.

[10] S. Babu, I. Ghosh, and B. S. Manoj, "Effort: A new metric for roadside unit placement in 5G enabled vehicular networks," in *Proc. IEEE 3rd 5G World Forum (5GWF)*, Bangalore, India, Sep. 2020, pp. 263–268.

[11] Y. Ni, J. He, L. Cai, J. Pan, and Y. Bo, "Joint roadside unit deployment and service task assignment for Internet of Vehicles (IoV)," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3271–3283, Apr. 2019.

[12] M. L. Balinski, "On finding integer solutions to linear programs," in *Proc. IBM scientific Symp. Combinat. Problems*. White Plains, NY, USA: IBM White Plains, 1966, pp. 225–248.

[13] A. Grasas, A. A. Juan, J. Faulin, J. de Armas, and H. Ramalhinho, "Biased randomization of heuristics using skewed probability distributions: A survey and some applications," *Comput. Ind. Eng.*, vol. 110, pp. 216–228, Aug. 2017.

[14] J. F. Stollsteimer, "The effect of technical change and output expansion on the optimum number, size and location of pear marketing facilities in a California pear producing region," Ph.D. dissertation, Univ. California Berkeley, Berkeley, CA, USA, 1961.

[15] G. Lee and A. T. Murray, "Maximal covering with network survivability requirements in wireless mesh networks," *Comput., Environ. Urban Syst.*, vol. 34, no. 1, pp. 49–57, Jan. 2010.

[16] A. J. Kadhim and S. A. H. Seno, "Maximizing the utilization of fog computing in internet of vehicle using SDN," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 140–143, Jan. 2019.

[17] S. A. Fernandez, D. Ferone, A. Juan, and D. Tarchi, "A simheuristic algorithm for video streaming flows optimisation with QoS threshold modelled as a stochastic single-allocation *p*-hub median problem," *J. Simul.*, pp. 1–14, Jan. 2021, doi: 10.1080/17477778.2020.1863754.

[18] K. Liu, X. Xu, M. Chen, B. Liu, L. Wu, and V. C. S. Lee, "A hierarchical architecture for the future internet of vehicles," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 41–47, Jul. 2019.

[19] Y. Ni, L. Cai, J. He, A. Vinel, Y. Li, H. Mosavat-Jahromi, and J. Pan, "Toward reliable and scalable internet of vehicles: Performance analysis and resource management," *Proc. IEEE*, vol. 108, no. 2, pp. 324–340, Feb. 2020.

[20] A. Abdrabou and W. Zhuang, "Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 1, pp. 129–139, Jan. 2011.

[21] C.-C. Lin and D.-J. Deng, "Optimal two-lane placement for hybrid VANET-sensor networks," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7883–7891, Dec. 2015.

[22] N. Nikookaran, G. Karakostas, and T. D. Todd, "Combining capital and operating expenditure costs in vehicular roadside unit placement," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7317–7331, Aug. 2017.

[23] M. Patra and C. S. R. Murthy, "Performance evaluation of joint placement and sleep scheduling of grid-connected solar powered road side units in vehicular networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 4, pp. 1197–1209, Dec. 2018.

[24] S. Anbalagan, A. K. Bashir, G. Raja, P. Dhanasekaran, G. Vijayaraghavan, U. Tariq, and M. Guizani, "Machine-learning-based efficient and secure RSU placement mechanism for software-defined-IoV," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13950–13957, Sep. 2021.

[25] D. Kim, Y. Velasco, W. Wang, R. N. Uma, R. Hussain, and S. Lee, "A new comprehensive RSU installation strategy for cost-efficient VANET deployment," *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 4200–4211, May 2017.

[26] H. Yu, R. Liu, Z. Li, Y. Ren, and H. Jiang, "An RSU deployment strategy based on traffic demand in vehicular ad hoc networks (VANETs)," *IEEE Internet Things J.*, early access, Sep. 10, 2021, doi: 10.1109/jiot.2021.3111048.

[27] A. Khezrian, T. D. Todd, G. Karakostas, and M. Azimifar, "Energy-efficient scheduling in green vehicular infrastructure with multiple roadside units," *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, pp. 1942–1957, May 2015.

[28] Y. Wang and J. Zheng, "Connectivity analysis of a highway with one entry/exit and multiple roadside units," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 11705–11718, Dec. 2018.

[29] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "A control and data plane split approach for partial offloading in mobile fog networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018, pp. 1–6.

[30] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Mobile edge computing partial offloading techniques for mobile urban scenarios," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018, doi: 10.1109/GLOCOM.2018.8647240.

[31] D.-T. Do, M.-S.-V. Nguyen, A.-T. Le, K. M. Rabie, and J. Zhang, "Joint full-duplex and roadside unit selection for NOMA-enabled V2X communications: Ergodic rate performance," *IEEE Access*, vol. 8, pp. 140348–140360, 2020.

[32] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Trans. Mobile Comput.*, early access, May 24, 2021, doi: 10.1109/tmc.2021.3082927.

[33] S. Vemireddy and R. R. Rout, "Clustering based energy efficient multi-relay scheduling in green vehicular infrastructure," *Veh. Commun.*, vol. 25, Oct. 2020, Art. no. 100251.

[34] L. Zhang and Y. Wang, "An offline roadside unit ON-OFF scheduling algorithm for energy efficiency of ad hoc networks," *IEEE Access*, vol. 6, pp. 59742–59751, 2018.

[35] L. D. C. Martins, D. Tarchi, A. A. Juan, and A. Fusco, "Agile optimization for a real-time facility location problem in internet of vehicles networks," *Networks*, early access, 2021, doi: 10.1002/net.22067.

[36] O. Kaiwartya, A. H. Abdullah, Y. Cao, A. Altameem, M. Prasad, C.-T. Lin, and X. Liu, "Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

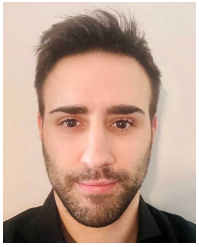
[37] L. V. Snyder, "Facility location under uncertainty: A review," *IIE Trans.*, vol. 38, no. 7, pp. 547–564, Jun. 2006.

[38] A. A. Juan, J. Faulin, R. Ruiz, B. Barrios, M. Gilibert, and X. Vilajosana, "Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem," in *Operations Res. Cyber-Infrastructure*, J. W. Chinneck, B. Kristjansson, and M. J. Saltzman, Eds. Boston, MA, USA: Springer, 2009, pp. 331–345.

[39] D. Mazza, A. Pages-Bernaus, D. Tarchi, A. A. Juan, and G. E. Corazza, "Supporting mobile cloud computing in smart cities via randomized algorithms," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1598–1609, Jun. 2018.

[40] A. A. Juan, C. G. Corlu, R. D. Tordecilla, R. de la Torre, and A. Ferrer, "On the use of biased-randomized algorithms for solving non-smooth optimization problems," *Algorithms*, vol. 13, no. 1, p. 8, Dec. 2019.

[41] A. A. Juan, J. Faulin, R. Ruiz, B. Barrios, and S. Caballé, "The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem," *Appl. Soft Comput.*, vol. 10, pp. 215–224, Jan. 2010.
[42] A. A. Juan, I. Pascual, D. Guimarans, and B. Barrios, "Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem," *Int. Trans. Oper. Res.*, vol. 22, no. 4, pp. 647–667, Jul. 2015.
[43] D. Ferone, A. Gruler, P. Festa, and A. A. Juan, "Enhancing and extending the classical GRASP framework with biased randomisation and simulation," *J. Oper. Res. Soc.*, vol. 70, no. 8, pp. 1362–1375, Aug. 2019.
[44] M. Marmol, L. D. C. Martins, S. Hatami, A. A. Juan, and V. Fernandez, "Using biased-randomized algorithms for the multi-period product display problem with dynamic attractiveness," *Algorithms*, vol. 13, no. 2, p. 34, Feb. 2020.
[45] A. A. Juan, J. Faulin, A. Ferrer, H. R. Lourenço, and B. Barrios, "MIRHA: Multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems," *TOP*, vol. 21, no. 1, pp. 109–132, Apr. 2013.
[46] L. D. C. Martins, C. Bayliss, P. J. Copado-Méndez, J. Panadero, and A. A. Juan, "A simheuristic algorithm for solving the stochastic omnichannel vehicle routing problem with pick-up and delivery," *Algorithms*, vol. 13, no. 9, p. 237, Sep. 2020.
[47] C. Bayliss, R. Guidotti, A. Estrada-Moreno, G. Franco, and A. A. Juan, "A biased-randomized algorithm for optimizing efficiency in parametric earthquake (Re) insurance solutions," *Comput. Oper. Res.*, vol. 123, Nov. 2020, Art. no. 105033.
[48] L. D. C. Martins, P. Hirsch, and A. A. Juan, "Agile optimization of a two-echelon vehicle routing problem with pickup and delivery," *Int. Trans. Oper. Res.*, vol. 28, no. 1, pp. 201–221, Jan. 2021.
[49] S. Zeadally, M. A. Javed, and E. B. Hamida, "Vehicular communications for ITS: Standardization and challenges," *IEEE Commun. Stand. Mag.*, vol. 4, no. 1, pp. 11–17, Dec. 2020.
[50] G. Yu, Q. Chen, and R. Yin, "Dual-threshold sleep mode control scheme for small cells," *IET Commun.*, vol. 8, no. 11, pp. 2008–2016, Jul. 2014.
[51] F. Han, S. Zhao, L. Zhang, and J. Wu, "Survey of strategies for switching off base stations in heterogeneous networks for greener 5G systems," *IEEE Access*, vol. 4, pp. 4959–4973, 2016.
[52] R. Martí, J. A. Lozano, A. Mendiburu, and L. Hernando, "Multi-start methods," in *Handbook of Heuristics*, R. Martí, P. M. Pardalos, and M. G. C. Resende, Eds. Cham, Switzerland: Springer, 2018, pp. 155–175.
[53] S. Ahn, C. Cooper, G. Cornuéjols, and A. Frieze, "Probabilistic analysis of a relaxation for the k-median problem," *Math. Oper. Res.*, vol. 13, no. 1, pp. 1–31, Feb. 1988.
[54] F. Barahona and F. A. Chudak, "Near-optimal solutions to large-scale facility location problems," *Discrete Optim.*, vol. 2, no. 1, pp. 35–50, Mar. 2005.



LUCA CESARANO was born in Naples, Italy, in 1993. He received the B.S. degree in computer engineering from the Università degli Studi di Napoli Federico II, Napoli, in 2018, and the M.Sc. degree in computer engineering from Alma Mater Studiorum—Università di Bologna, Bologna, Italy, in 2021. From September 2020 to February 2021, he was a Visiting student working to the final thesis with the Universitat Oberta de Catalunya, Barcelona, Spain. His research interests include the operations research, transportation and logistics, artificial intelligence, cloud computing, and distributed systems fields.

ANDREA CROCE was born in Naples, Italy, in 1994. He received the B.S. degree in computer engineering from the University of Naples Federico II, Napoli, in 2018, and the M.Sc. degree in computer engineering from Alma Mater Studiorum—University of Bologna, Bologna, Italy, in 2021. From September 2020 to February 2021, thanks to Erasmus+ Program, he worked with the DPCS-ICSO Research Group, IN3-EIMT, Universitat Oberta de Catalunya (UOC), Barcelona, Spain, for the preparation of the final thesis. His research interests include operational research, transportation and logistics, and heuristics algorithms for internet of vehicles scenario.

LEANDRO DO CARMO MARTINS received the B.Sc. and M.Sc. degrees in computer science from the Federal University of Ouro Preto, Brazil, and the Ph.D. degree in network and information technologies from the ICSO Research Group, Universitat Oberta de Catalunya, Barcelona, Spain. His research interests include operational research, combinatorial optimization, transportation and logistics, metaheuristics, biased-randomized algorithms, agile optimization, and simulation.

DANIELE TARCHI (Senior Member, IEEE) was born in Florence, Italy, in 1975. He received the M.Sc. degree in telecommunications engineering and the Ph.D. degree in informatics and telecommunications engineering from the University of Florence, Florence, Italy, in 2000 and 2004, respectively.

From 2004 to 2010, he was a Research Associate with the University of Florence, Italy. From 2010 to 2019, he was an Assistant professor with the University of Bologna, Bologna, Italy, where he has been an Associate Professor, since 2019. He is the author of around 130 published articles in international journals and conference proceedings. He has been involved in several national and international research projects, and collaborates with several foreign research institutes. His research interests include wireless communications and networks, satellite communications, edge computing, fog computing, smart cities, and optimization techniques.

Prof. Tarchi is an Editorial Board Member of IEEE WIRELESS COMMUNICATIONS LETTERS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and *IET Communications*. He has been the Symposium Co-Chair of IEEE WCNC 2011, IEEE Globecom 2014, IEEE Globecom 2018 and IEEE ICC 2020, and the Workshop Co-Chair at IEEE ICC 2015.

ANGEL A. JUAN received the M.Sc. degree in mathematics and the Ph.D. degree in industrial engineering. He completed a Predoctoral Internship with Harvard University and Postdoctoral Internships with the Massachusetts Institute of Technology and the Georgia Institute of Technology. He is currently a Full Professor in operations research & industrial engineering with the Computer Science Department, Universitat Oberta de Catalunya, Barcelona, Spain. He is also the Director of the ICSO Research Group. He has published over 120 articles in JCR-indexed journals and over 275 papers indexed in Scopus. His research interests include applications of simulation, metaheuristics, and machine learning methods in different fields, including logistics and transportation, finance, and telecommunication systems.