WILEY | Hindawi

*Research Article*

# LSTM-Based Traffic Load Balancing and Resource Allocation for an Edge System

**Thembelihle Dlamini** [iD][1] **and Sifiso Vilakati** [iD][2]

[1]*Department of Electrical and Electronic Engineering, University of Eswatini, Kwaluseni, Eswatini*
[2]*Department of Statistics and Demography, University of Eswatini, Kwaluseni, Eswatini*

Correspondence should be addressed to Thembelihle Dlamini; tldlamini@uniswa.sz

The massive deployment of small cell Base Stations (SBSs) empowered with computing capabilities presents one of the most ingenious solutions adopted for 5G cellular networks towards meeting the foreseen data explosion and the ultralow latency demanded by mobile applications. This empowerment of SBSs with *Multi-access Edge Computing* (MEC) has emerged as a tentative solution to overcome the latency demands and bandwidth consumption required by mobile applications at the network edge. The MEC paradigm offers a limited amount of resources to support computation, thus mandating the use of intelligence mechanisms for resource allocation. The use of green energy for powering the network apparatuses (e.g., Base Stations (BSs), MEC servers) has attracted attention towards minimizing the carbon footprint and network operational costs. However, due to their high intermittency and unpredictability, the adoption of learning methods is a requisite. Towards intelligent edge system management, this paper proposes a Green-based Edge Network Management (GENM) algorithm, which is an online edge system management algorithm for enabling green-based load balancing in BSs and energy savings within the MEC server. The main goal is to minimize the overall energy consumption and guarantee the Quality of Service (QoS) within the network. To achieve this, the GENM algorithm performs dynamic management of BSs, autoscaling and reconfiguration of the computing resources, and on/off switching of the fast tunable laser drivers coupled with location-aware traffic scheduling in the MEC server. The obtained simulation results validate our analysis and demonstrate the superior performance of GENM compared to a benchmark algorithm.

## 1. Introduction

The foreseen *dense* deployment of BS empowered with computing capabilities in order to meet the ultralow latency demanded by mobile users raises concerns related to energy consumption. Apart from the fact that BS energy costs account for a large part of the operating expenses of MN operators, there are also increasing concerns regarding their environmental impact in terms of high carbon dioxide ($CO_2$) emissions. In an effort to minimize energy consumption and energy costs in 5G cellular networks within the MEC paradigm, this paper advocates for the integration of EH systems into network apparatuses and the use of container-based virtualization within computing platforms (i.e., MEC servers). The use of green energy mitigates the negative environmental

impact of MN and enables cost-saving for mobile operators in terms of lowering operational energy costs. The motivation towards green energy is due to the fact that current trends in battery and solar module costs show a reduction in prices. The benefits of container-based virtualization is the reduction in energy drained in the computing platform due to their lower overheads when compared with VM [1, 2]. For a qualitative comparison of different virtualization techniques, interested readers are referred to [1].

In this paper, a group of EH BS placed in proximity to a EH-MEC server is considered as an *edge system*. The EH-MEC server manages the BS system, deciding upon the allocation of transmission resources and also handling the computing and communication processes. In general, renewable energy systems are dimensioned to guarantee

the autonomous operation of the edge system. Thus, it is desirable that the utilization of green energy be made one of the performance metrics when designing load-balancing strategies [3, 4], instead of the network impact [5, 6]. As a result, the green-based load balancing is a promising technique for optimizing MEC performance since it exploits the spatial diversity of the available green energy to reshape the network load among the BSs [4]. In this case, MN can prioritize the utilization of BS with sufficient green energy to serve more traffic while reducing the traffic loads for those BS consuming on-grid power. For instance, in the MEC server, a trade-off between the green energy utilization and the amount of workload that can be computed locally should be carefully evaluated.

In this regard, it is worth noting that the energy consumption within the virtualized computing platform is due to (i) the active computing resources, i.e., VM or containers [7–9], and (ii) the network communications, communication related to transmission drivers [10, 11], and the intracommunications [12]. In order to alleviate this, this paper assumes that the container-based virtualization be deployed in the MEC server as containers are lightweight, i.e., demand less memory space, have shorter start-up time, and offer software portability. At each time instance, the containers are provisioned based on the forecasted server workloads, a technique referred to as *autoscaling*. In addition, the transmission drivers used for data transfers within the server are tuned by taking into account the mobile user's location. The server is also capable of caching the frequently requested contents locally.

In densely deployed BS, the energy drained is due to the always-on design approach [5, 6, 13], yet traffic load varies during the day, e.g., the demand is low during the night. Therefore, in order to intelligently manage the BS towards minimizing the energy consumption, the green-based load balancing technique is employed, i.e., BS sleep modes are enabled in some BS using green energy as a performance metric.

*1.1. Paper Contributions.* This paper considers an energy cost model that takes into account the computing, caching and communication processes within the MEC server, and transmission-related energy consumption in BS. Here, the GENM algorithm is proposed for enabling *green-based* traffic load balancing, i.e., the BS are dynamically switched on/off based on their harvested energy, autoscaling, and reconfiguring the computing resources, and the tuning of transmission drivers. This entails using a minimum number of optical drivers for real-time data transfers, over a short-term horizon. In order to solve the energy consumption minimization problem, the GENM algorithm performs online supervisory control, utilizing the learned traffic load and the harvested energy patterns. Then, the output is utilized within a LLC policy [14] to obtain the optimal system control actions that yield the desired energy savings that guarantee the required QoS.

This work is an extension of [11], where energy savings and QoS guarantee were considered only within a virtualized computing platform placed in proximity to a cluster of BS. In [11], the role of the MEC server is to handle the offloaded computational workload only, which means that the energy cost model lacks the consideration of the BS management procedures, caching process, and the use of containers.

The summary of contributions are listed as follows:

(i) The use of container-based virtualization is introduced as they are lightweight, i.e., demand less memory space, have shorter start-up time, and offer software portability

(ii) The proposed GENM algorithm, which is an online edge management system, makes use of predictive optimization, specifically using the LLC, where green-based load balancing, container provisioning, and the tuning of the transmission drivers is performed based on the learned information

(iii) The numerical results, obtained with real-world harvested energy and traffic load traces, show that the proposed optimization strategy is able to efficiently manage the edge network resources in order to minimize the energy drained under the guidance of the intelligent online-based resource manager and the energy-saving procedures

In order to achieve these, the remainder of the paper is organized as follows: Section 2 describes the related work. Section 3 explains the system model. In Section 4, the design and the implementation of the online algorithm are presented. Simulated results are discussed in Section 5. Lastly, the work is concluded in Section 6.

## 2. Related Work

*2.1. Methods for Load Balancing in MN.* Load balancing has been studied towards data center management whereby the data center servers employ temporal dependency strategies, i.e., the servers are turned on/off depending on the arrival rates of workloads. This significantly differs from our considered problem as we consider load balancing in SBSs powered by green energy. Towards load balancing, the dynamic BS switching on/off strategies have been used. However, this may have an impact on the network due to the load that is offloaded to the neighboring BS(s). To avoid this, the BS to be switched off is carefully identified within the BS cluster. In [5, 6], the *network impact* is used to identify the BS to be switched off, one at a time, with no significant network performance degradation. Taking into account daily traffic load variation, strategies for opportunistic utilization of the unexploited third-party small cell (SC) capacity is exploited towards energy savings in [15], in order to enable the switching off of some BSs. Here, an offloading mechanism is introduced, where the operators lease the capacity of a SC network owned by a third party in order to switch off their BS (Macro BS) and maximize their energy efficiency, when the traffic demand is low.

The use of green energy as a performance metric has been explored within the Radio Access Network (RAN) [3, 16]. Along the lines of MN softwarization, a distributed user association scheme that makes use of the SoftRAN concept for traffic load balancing via the RAN Controller (RANC) is proposed in [3]. Here, the user association algorithm runs on the RANC, and the users report their downlink data rates via the associated BS to the RANC, where the traffic loads

from individual users and BS are measured. The algorithm enhances the network performance by reducing the average traffic delivery latency in BS as well as to reduce the on-grid power consumption by optimizing the green energy usage. Then, the authors in [16] proposed to optimize the utilization of green energy for cellular networks by optimizing the BS transmission power. The proposed scheme achieves significant on-grid power savings by scheduling the green energy consumption along the time domain for individual BS and balancing the green energy consumption among the BSs.

Along the lines of MEC, the authors in [4] proposed a framework for jointly performing load balancing, admission control, and energy purchase within a network of EH-powered BS with the goal of minimizing the computation delay and data traffic drops (i.e., increasing the locally computed workloads). This work uses green energy as a performance metric. To solve this problem, an online and distributed algorithm is proposed leveraging the Lyapunov optimization with the perturbation technique. Here, the algorithm makes the traffic load decisions without forecasting the future traffic load and harvested energy. In contrast, the work presented in this paper considers the short-term future traffic load and the harvested energy for decision making. Then, in our previous work [6], a supervisory online control algorithm that makes use of clustering and the network impact metric towards load balancing in MN is proposed. Here, the BSs are empowered with computation capabilities (with VM as computing resources), the LSTM neural network is used for forecasting, and the LLC policy handles foresighted optimization. Even though these works perform load balancing, the problem and scenario considered in this paper are different. Here, a MEC server manages the SBSs powered by green energy. Similar to [11], forecasting and foresighted optimization are used for edge system management.

Load balancing that follows the energy routing, i.e., more computational workload is offloaded to where more energy is available, is presented in [17]. To handle spatial uneven computation workloads experienced by the MEC-enabled BSs, the authors proposed a peer offloading scheme. Here, the BSs share their computing resources and energy costs.

## 2.2. Methods for Energy Saving within Computing Platforms.
Green computing over data centers is an emerging paradigm that aims at performing the dynamic energy-saving management of data center infrastructures. Here, procedures for the dynamic on/off switching of servers have been proposed as a way of minimizing energy consumption in computing platforms. A novel postdecision state-based learning algorithm for server provisioning at the network edge is presented in [18]. This work incorporates green energy. At the beginning of the time slot, the servers are consolidated, i.e., the number of turned on physical servers are minimized, using the learned optimal policy for dynamic workload offloading and the autoscaling (or right-sizing). Then, in our previous works [6, 11, 19], VM *soft-scaling* (i.e., the reduction of computing resources per time instance) is employed towards energy saving in virtualized platforms either energized by only renewable energy or hybrid supplies (solar and power grid). This is achieved by forecasting the traffic load and har-

vested energy and then employing foresighted optimization to obtain the system control inputs. The work of [20] uses an iterative algorithm to obtain the number of computing resources (VMs) to be provisioned within a node that transmits to clients wireless. Then, the work of [21] considers a vehicular scenario where vehicles connect wireless to fog nodes and then develop an adaptive scheduler, which computes on-the-fly the solutions of both the resource reconfiguration and consolidation problems. For this purpose, the primal-dual algorithm is used.

In computing platforms, computation offloading strategies can be jointly exploited together with delay constraints towards energy savings. The authors in [22] proposed an offloading policy to find the optimal place where to offload and the amount of offloaded task data. In this work, the time taken for processing the offloaded task is reduced, at the same time consuming less energy. Then, in [23], an efficient scheduling for latency-sensitive applications is proposed towards energy and response time minimization. The achieved results show a reduction in delay and network usage and the energy consumption. In addition, the works of [24] jointly optimize the computing and communication resources, taking into account the local task execution delay and transmission delay. To meet the task delay requirements, in [25], the heterogeneous clouds, i.e., edge and remote cloud, are coordinated. Here, different policies are employed in the clouds. In this, the edge cloud handles tasks with loose delay bounds and drops tasks with stringent delay bounds when the traffic load is heavy.

Towards minimizing energy consumption induced by communication activities within a computing node, the idea of tuning transmission drivers, as one of the energy-saving strategies within the MN infrastructure, is first conceived in [10, 26] where a computing node (router) is considered. Here, it is observed that having the least number of data transmission drivers (*fast* tunable lasers) can yield a significant amount of energy savings. Motivated by the aforementioned works, within the MEC paradigm, the authors in [11] put forward a traffic engineering- and MEC LS-based algorithm that uses a location-aware procedure for provisioning the transmission drivers for data transfer towards target BS. Here, the MEC LS API is employed for retrieving the UE's location and then passing the information to the authorized applications within the MEC platform, for decision-making.

Overall, these works numerically analyze and test the energy performance of some state-of-the-art schedulers for computing platforms, but do not attempt to optimize it through the dynamic joint scaling of the available communication-plus-computing resources. The joint analysis of the computing-plus-communication energy consumption within the MEC paradigm is still an open research topic.

## 2.3. Methods for Guaranteeing Quality of Service and Enabling Energy Savings (within the MEC Paradigm).
The mobile operator's goal is to provide QoS Internet services for large populations of clients, while minimizing the overall computing-plus-communication energy consumption. Hence, a trade-off is required between QoS and energy savings. Future MNs are expected to learn the diverse characteristics of user behavior, as well as renewable energy variations, in

order to autonomously determine good system configurations. Towards this goal, online forecasting using ML techniques and the LLC method can yield the desired system behavior when taking into account the environmental inputs, i.e., BS traffic load, server workloads, and energy to be harvested. Next, the mathematical tools that are used in this research work are reviewed, namely, the LLC method [14, 27, 28] and LSTM neural networks [29, 30].

The LLC has been used in [27] to address a resource provision problem within virtualized environments. The optimization problem is posed as a profit maximization problem under uncertainty and the LLC formulation models the cost of control. To address the uncertainty over the workload arrival, the Kalman filter is used. Then, in [31], an online supervisory control scheme based on LLC policies is proposed. Here, after the occurrence of an event, the next control action is determined by estimating the system behavior a few steps into the future using the currently available information as inputs. The control action exploration is performed using a search tree assuming that the controller knows all future possible states of the process over the prediction horizon. An online control framework for resource management in switching hybrid systems is proposed in [14], where the system's control inputs are finite. The relevant parameters of the operating environment, e.g., workload arrival, are estimated and then used by the system to forecast future behavior over a look-ahead horizon. From this, the controller optimizes the predicted system behavior following the specified QoS through the selection of the system controls. In [6, 19], a LLC-based supervisory algorithm is proposed to obtain the system control actions yielding the desired trade-off between energy consumption and QoS. Here, the traffic load and harvested energy are forecasted and then used as input in the algorithm. The BS are densely deployed in [6], and each BS is empowered with computation capabilities. Then, in [19], a remote site powered by only green energy is considered.

LSTM can be used for multistep time series forecasting as it is able to handle the long-term dependencies due to its inherent capability of storing past information and then recalling it. The forecasting method is utilized in [6, 19] within an LLC-based algorithm to obtain the system control actions yielding the desired trade-off between energy consumption and QoS. The application of the LSTM network is extended to include ITS in [32]. A new ITS edge analytics architecture that makes use of deep learning techniques that run either on the mobile devices or on the intravehicle processors for data analytics is presented. A combination of LSTM networks and deep CNN is adopted, i.e., CNN-LSTM network, for path selection in autonomous vehicles, whereby the CNN is used for feature extraction, and then, the extracted information is fed into LSTM networks for driving path selection. Forecasting server workloads using the LSTM network can be beneficial for dynamic resource scaling and power consumption in cloud computing data centers. In [33], a forecasting model using the LSTM network for predicting future data center workloads is proposed, and then, the results are fed into the resource manager for decision-making, which involves either scaling up or down the computing resources (servers in this case).

## 3. System Model

In line with ETSI-proposed MEC deployment scenarios discussed in [34], the considered network scenario is illustrated in Figure 1 where the proposed model is cache-enabled, TCP/IP offload capable (i.e., enables *partial* offloading in the server's NIC such as checksum computation [35]). The virtualized MEC server is assumed to be hosting $C$ containers deployed at an aggregation point, which is in proximity to a cluster of $N$ BS from the same MN operator. The BS are interconnected to the MEC server for computation workload offloading. Each network apparatus (BS, MEC server) is mainly powered by renewable energy harvested from wind and solar radiation, and it is equipped with an EB for energy storage. In this case, energy can only be purchased from the grid supply to supplement the renewable energy supplies. The BS coverage areas overlap so that green-based load balancing is possible. The EM is an entity responsible for selecting the appropriate energy source to fulfill the EB and also for monitoring the energy level of the EB. Then, the electromechanical switch (SW) aggregates the energy sources to fulfill the EB level. In the MEC server, there is the presence of a virtualized access control router which acts as an access gateway for admission control, responsible for local and remote routing, and it is locally hosted as an application. Also, the MEC platform is able to track user location via the MEC Location Service Application Programmable Interface (LS API). Lastly, a discrete-time model is considered whereby time is discretized as $t = 1, 2, \cdots$ and each time slot $t$ has a fixed duration $\tau = 30$. The list of symbols that are used in the paper is reported in Table 1.

*3.1. Communication Traffic and Energy Consumption.* From a networking perspective, the understanding and characterization of energy consumption within the MN can pave the way towards more efficient and user-oriented networking solutions. This can be achieved through the use of historical mobile traffic traces such as Call Detail Records (CDRs) obtained from mobile operators, specifically in the EPC network. Due to the difficulties in obtaining relevant open-source datasets containing computing requests, real MN traffic load traces obtained from the TIM network (availed through the Big Data Challenge [36]) are used to emulate the computational load. In order to understand the daily traffic load patterns, the clustering algorithm X-means [37] has been applied to classify the load profiles into several categories. Here, each BS $n$ is assumed to have a related load profile $L_n(t)$ which is picked at random as one of the four clusters in Figure 2. In addition, it is assumed that $L_n(t)$ consists of 80% delay sensitive workloads $\gamma_n(t)$ and the remainder is delay-tolerant workloads. The total aggregate delay-sensitive workload per time instance is $\xi(t) = \sum_{n=1}^{N} \gamma_n(t)$.

The virtualized router in the MEC server of Figure 1 determines the amount of workload that can be accepted by the input buffer at slot $t$, and the aggregated (or admitted) workload is denoted by $L_{\text{in}}(t) \in [0, L_{\text{in}}^{\max}]$ (measured in [Mbits]). $L_{\text{in}}^{\max}$ is the maximum input buffer size. In addition, it is assumed that the input/output (I/O) queues of the MEC server are loss-free, and they implement the First-In First-
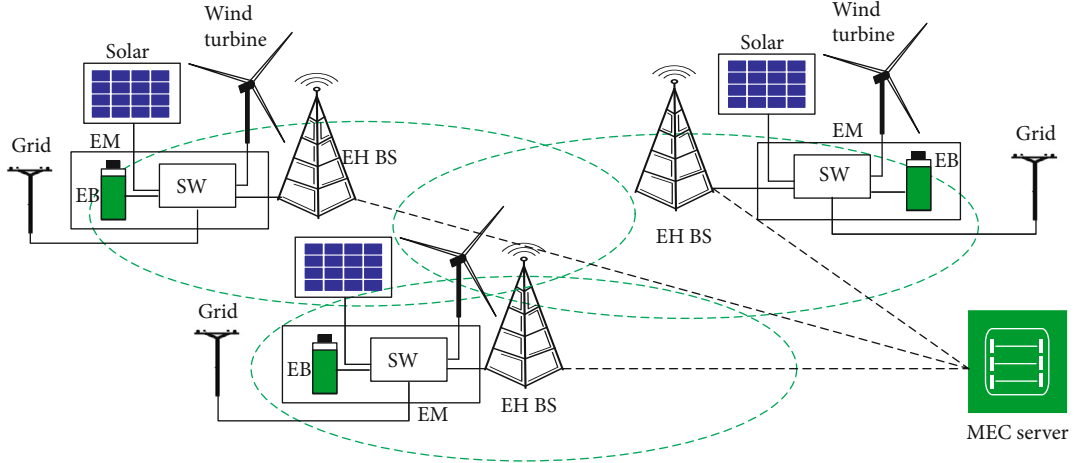
FIGURE 1: Edge system energized by hybrid energy sources: on-grid power and green energy (solar and wind).

Out (FIFO) service discipline, thus $L_{in}(t) = L_{out}(t)$, where $L_{out}(t)$ is the amount of the aggregate computed workload at the output buffer.

The total energy consumption ([J]) for the edge system at time slot $t$ is formulated as follows, inspired by [11, 12]:

$$\theta_{EDGE}(t) = \theta_{COMM}(t) + \theta_{MEC}(t), \quad (1)$$

where $\theta_{COMM}(t)$ is the energy consumption term induced by all BS communications and $\theta_{MEC}(t)$ is the energy consumption term induced by the MEC server's computing, caching, and communication processes.

*3.1.1. BS Energy Cost.* The overall energy consumption within the coverage area is defined as the sum of all the BS components:

$$\theta_{COMM}(t) = \sum_{n=1}^{N} \theta_{BS,n}(t) = \sum_{n=1}^{N} (\delta_n(t)\theta_0 + \theta_{load,n}(t)), \quad (2)$$

where $\delta_n(t) \in \{0, 1\}$ is the BS $n$ switching status indicator (1 for *active mode* and 0 for *power saving mode*); $\theta_0$ is a constant value (load independent) representing the operation energy which includes baseband processing, radio frequency power expenditures, etc.; $\theta_{load,n}(t)$ is the load-dependent BS transmission power to the served users that guarantees low latency at the edge. It is obtained by using the transmission model in [38].

*3.1.2. MEC Energy Cost.* The energy drained due to the computing, caching, and communication processes is defined as:

$$\theta_{MEC}(t) = \theta_{CNT}(t) + \theta_{SWT}(t) + \theta_{OFF}(t) + \theta_{LNK}(t) + \theta_{DR}(t) + \theta_{CC}(t), \quad (3)$$

where $\theta_{CNT}(t)$ is the energy drained due to the active containers, w.r.t CPU utilization, and $\theta_{SWT}(t)$ is the energy drained due to containers adapting to new processing rates $f_c(t) \in [f_0, f_{max}][(Mbit/s)]$. The term $f_0$ is the zero speed of the container, e.g., deep sleep or shutdown, and $f_{max}$ is the maximum available processing rate for container $c$. It is

worth noting that actual containers are generally instantiated atop physical computing cores which offer only a finite set of processing speeds. The term $\theta_{OFF}(t)$ is the energy induced by the TCP/IP offload on the NIC, and $\theta_{LNK}(t)$ is the energy drained due to the virtual-links (to-and-from containers) communication cost. Then, $\theta_{DR}(t)$ is the amount of energy consumed by the active transmission drivers, and $\theta_{CC}(t)$ is the total energy cost incurred by the content caching process.

In this regard, it is assumed that real-time processing of computation workloads are performed in parallel over the containers interconnected by a rate-adaptive Virtual LAN (VLAN). In addition, it is also assumed that the CPU frequency is fixed at each user and may vary over users. The amount of energy consumed by the CPU is related to the provisioned computing resources, i.e., the CPU share allocated to each container, per time instance $t$, named $C(t) \leq C$, index by $c$. Thus, $\theta_{CNT}(t)$ is defined as [19]:

$$\theta_{CNT}(t) = \sum_{c=1}^{C(t)} \theta_{idle,c}(t) + \psi_c(t)(\theta_{max,c}(t) - \theta_{idle,c}(t)), \quad (4)$$

where $\theta_{idle,c}(t)$ represents the *static* energy drained by container $c$ in the idle state, $\psi_c(t) = (f_c(t)/f_{max})^2$ is the utilization function of container $c$ [28], and $\theta_{max,c}(t)$ is the *maximum* energy that container $c$ can consume. The quantity $\psi_c(t)(\theta_{max,c}(t) - \theta_{idle,c}(t))$ represents the *dynamic* energy component of container $c$.

The intelligent resource manager implements a suitable frequency-scaling policy in real-time, in order to allow the containers to scale up/down their processing rates $f_c(t)$ at the minimum cost. In this regard, it should be noted that switching from the processing frequency $f_c(t-1)$ (the processing rate at the $(t-1)$ time instance) to the next processing frequency $f_c(t)$ entails an energy cost, $\theta_{SWT}(t)$. This depends on the absolute processing rate gap $|f_c(t) - f_c(t-1)|$; thus, $\theta_{SWT}(t)$ is defined as:

$$\theta_{SWT}(t) = \sum_{c=1}^{C(t)} z_e(f_c(t) - f_c(t-1))^2, \quad (5)$$

TABLE 1: Notation: list of symbols used in the analysis.

| Symbol | Description |
|---|---|
| **Input parameters** | |
| $C$ | Maximum number of containers hosted by the MEC server, indexed by $c$ |
| $N$ | Number of BS, indexed by $n$ |
| $\xi(t)$ | Aggregate computational workload |
| $\tau$ | Time slot duration |
| $L_{in}(t)$ | Amount of aggregate workload at the input buffer |
| $L_{out}(t)$ | Amount of aggregate workload at the output buffer |
| $L_{out}^{max}(t), L_{in}^{max}(t)$ | Workload buffers maximum capacity |
| $f_{max}$ | Maximum processing rate for container $c$ |
| $\theta_{idle,c(t)}$ | Static energy consumed by container $c$ in the idle state |
| $\theta_{max,c(t)}$ | Maximum energy consumed by container $c$ at maximum processing rate |
| $z_e$ | Per-container reconfiguration cost caused by a unit-size frequency switching |
| $\lambda_c(t)$ | Workload fraction to be computed by the $c$-th container |
| $\lambda_{max}$ | Maximum computation load per container |
| $\Delta$ | Maximum per-slot and per-container allowed processing time |
| $\theta_{idle}^{NIC}(t)$ | Energy drained by the NIC when powered, with no data transfer |
| $M$ | Maximum number of multiple fast tunable lasers |
| $\beta_{max}$ | Maximum energy buffer capacity |
| $\beta_{up}, \beta_{low}$ | Upper and lower energy buffer thresholds |
| **Variables** | |
| $\theta_{COMM}(t)$ | Total BS energy consumption at time slot $t$ |
| $\theta_{MEC}(t)$ | Server's energy consumption at time slot $t$ |
| $\theta_{CNT}(t)$ | Energy drained due to the active containers, w.r.t CPU utilization, at time slot $t$ |
| $\theta_{SC}(t)$ | Energy drained due to container switching the processing rates at time slot $t$ |
| $\theta_{OFF}(t)$ | Energy induced by the TOE at $t$ |
| $\theta_{LNK}(t)$ | Energy drained due to the virtual-link communication cost at time slot $t$ |
| $\theta_{DR}(t)$ | Energy drained due to the number of active transmission drivers at time slot $t$ |
| $\theta_{CC}(t)$ | Total energy cost incurred by the content caching process |
| $C(t)$ | Number of containers to be active in time slot $t$ |
| $f_c(t)$ | Instantaneous processing rate |
| $\psi_c(t)$ | Load-dependent factor |
| $r_c(t)$ | $c$-th virtual link communication rate at slot $t$ |
| $\zeta_n(t)$ | BS switching status indicator at $t$ |
| $\theta_{max}^{NIC}(t)$ | Maximum energy drained by the TOE at $t$ |
| $\chi_t(t)$ | The expected processing time |

TABLE 1: Continued.

| Symbol | Description |
|---|---|
| $M(t)$ | Number of active transmission drivers at $t$ |
| $b(t)$ | Energy buffer level in slot $t$ |
| $H(t)$ | Harvested energy profile in slot $t$ |
| $E(t)$ | Purchased grid energy in slot $t$ |

where $z_e$ is the per-container reconfiguration cost caused by a unit-size frequency switching. Typically, $z_e$ is limited to a few hundreds of millijoule per square megahertz.

Before proceeding, it is worth noting the following: at the beginning of the time slot $t$, the online algorithm adaptively allocates the available resources and then determine the containers that are demanded, $C(t)$; the size of the workload allocated to the container $c$, denoted by $\lambda_c(t)$; and $f_c(t)$ for container $c$ that will yield the desired or expected processing time, $\chi_c(t) = \lambda_c(t)/f_c(t)$. $\chi_c(t) \leq \Delta$, where $\Delta$ is the maximum per-slot and per-container processing time ([s]). Note that $L_{in}(t) = \sum_{c=1}^{C(t)} \lambda_c(t)$ is the amount of computational workload admitted in the MEC server, by the router. The amount of the workload to be admitted per slot shall be decided at the beginning of each time slot depending on the forecasted green energy, grid power to be purchased, and the expected computational workloads $\hat{L}_{in}(t)$. Moreover, virtualization technologies specify the minimum and maximum amount of resources that can be allocated per container [40]; thus, the maximum amount is denoted by $\lambda_{max}$. Lastly, the container(s) provisioning and workload allocation is discussed in Section 4.2, Remark 1, and $f_c(t) = {}^\Delta \lambda_c(t)/\Delta$.

By implementing a TCP Offload Engine (TOE) in high-speed computing environments, some TCP/IP processing is offloaded to the network adapter for the purpose of reducing the CPU utilization. To obtain the energy cost incurred, the performance measure for the Broadcom (Fibre) 10 Gbps NIC [35] is considered here as an example of a TCP/IP offload-capable device. Note that $\theta_{OFF}(t)$ is data volume-dependent and it is obtained as:

$$\theta_{OFF}(t) = \zeta(t)\,\theta_{idle}^{NIC}(t) + \theta_{max}^{NIC}(t), \tag{6}$$

where $\theta_{idle}^{NIC}(t) > 0$ is the energy drained by the TOE when powered, with all links connected without any data transfer. This provides an opportunity for switching off the network adapter if there is no data transfer, making the energy drained to be zero. For this, $\zeta(t) = (0, 1)$ is the switching status indicator (1 for active state and 0 for idle state) and $\theta_{max}^{NIC}(t) = (g(t) \cdot L_{in}(t))/\eta$ is the maximum energy drained, where $g(t)$ is a fractional value representing the amount of load computed in the network adapter and $\eta$ is the NIC best throughput performance, hereby obtained as a fixed value measured in [Gbit/].

In order to keep the transmission delays from (to) the scheduler to (from) the connected containers at a minimum value, it is assumed that each container $c$ communicates with the resource scheduler through a dedicated reliable link that
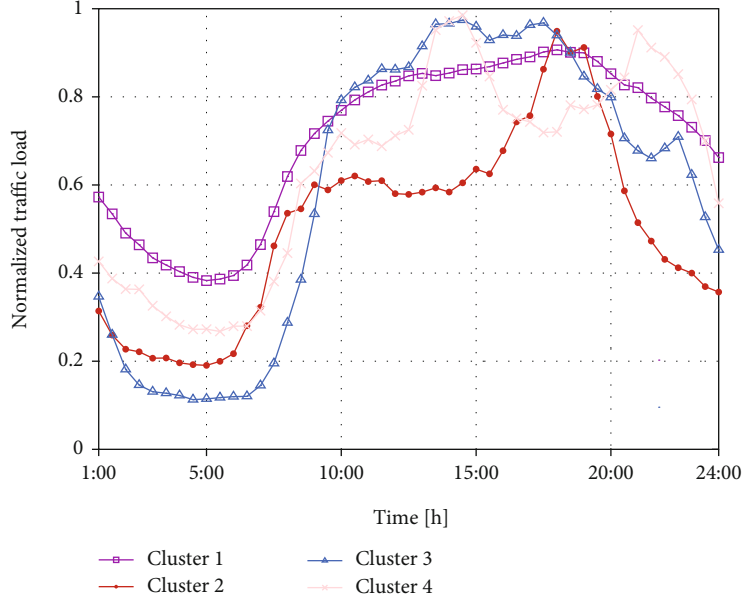
FIGURE 2: Normalized BS traffic load behavior represented as clusters. The data from [39] has been split into four representative clusters.

operates at the transmission rate of $r_c(t)$ [(bit/s)]. Thus, the energy needed for sustaining the two-way $c^{th}$ link is defined, inspired by [41]:

$$\theta_{LNK}(t) = 2 \sum_{c=1}^{C(t)} P_c(r_c(t))(\lambda_c(t)/r_c(t)), \tag{7}$$

where $P_c(r_c(t)) = S_c(2^{r_c(t)/W_c} - 1)$ is the power drained by the $c^{th}$ communication link and $S_c = (W_c \times N_0^{(c)})/g_c$. $N_0^{(c)}$ (W/Hz) is the noise spectral power density, $W_c$ is the bandwidth, and $g_c$ is the (nonnegative) gain of the $c^{th}$ link. In practical application scenarios, the maximum per-slot communication rate within the intra-VLAN is generally limited up to an assigned value $r_{max}$. Thus, the following hard constraint must hold: $\sum_{c=1}^{C(t)} r_c(t) \leq r_{max}$.

In this regard, a two-way per-task execution delay is considered. Here, there is a total of $c = \{1, \cdots, C(t)\}$ link connection delays, each denoted by $\rho_c(t) = \lambda_c(t)/r_c(t)$, and $\chi_c(t) \leq \Delta$ where $\Delta$ is the server's response time, i.e., the maximum time allowed for processing the total computation load, and it is fixed in advance regardless of the task size allocated to container $c$. Since parallel real-time processing is assumed in this work, the overall communication equates to $2\rho_c(t) + \Delta$. Therefore, the hard per-task delay constraint on the computation time is: $\max\{2\rho_c(t)\} + \Delta = \tau_{max}$, where $\tau_{max}$ is the maximum tolerable delay, which is fixed in advance.

Edge distributed devices utilize low-level signaling for information sharing. Thus, edge computing systems receive information from mobile devices within the local access network to discover their location. In return, for every *client* who offloaded their task into the MEC server associated with the radio nodes, i.e., BS, its location and the computation result are known through the LS (which is a service that supports UE location retrieval mechanism, and then passing the infor-

mation to the authorized applications within the server), thus enabling the location-aware traffic routing and obtaining the number of transmission drivers to be active for data transfers. The term $\theta_{DR}(t)$ depends on the number of active laser (optical) drivers, named $M(t) \leq M$, where $M$ is the total number of drivers that are required for transferring $\ell_m(t) \in L_{out}(t)$ in time slot $t$ ($\ell_m(t)$ is the downlink traffic volume ([bits] of the driver at slot $t$). $L_{out}(t)$ is accumulated over a fixed period of time to form a *batch* at the output buffer. This means that a large number of drivers yield large transmission speed while at the same time resulting in high energy consumption [26]. Therefore, the energy consumption can be minimized by launching an optimal number of drivers for the data transfer.

The energy drained during the data transmission process consists of the following: a constant energy for utilizing each fast tunable driver denoted by $d_m(t)([J/s])$, the target transmission rate $r_0$ [bits/s], and $L_{out}(t)$. Thus, the energy is inspired by [11]:

$$\theta_{DR}(t) = \sum_{m=1}^{M(t)} \frac{d_m(t)\, l_m(t)}{r_0}, \tag{8}$$

where the parameter $M(t)$ is obtained using the total number of target BS as $M(t) = \lceil 1/u \cdot ((\omega(t) + 1)/(\omega(t)))^2 \rceil$, where $\omega(t) = \sqrt{\rho/(\sigma N_{BS}(t))} \cdot \in (0, 1]$ is a controllable factor that determines the delay constraint of optical networks, $\sigma$ ([ms]) is the reconfiguration cost for tuning the transceivers, $N_{BS}(t)$ is an integer value representing the total number of target BS at time slot $t$, and $\rho$ is the number of time slots at which the computed workload is accumulated at the output buffer. Thus, the terms $\sigma$ and $\rho$ are fixed values, and $L_{out}(t)$ is equally distributed over the $M(t)$ drivers.

The MEC server is able to cache contents from the Internet and store the contents closer to mobile users. The caching process also contributes to the energy consumption in the server. The caching process is restricted to only viral content.
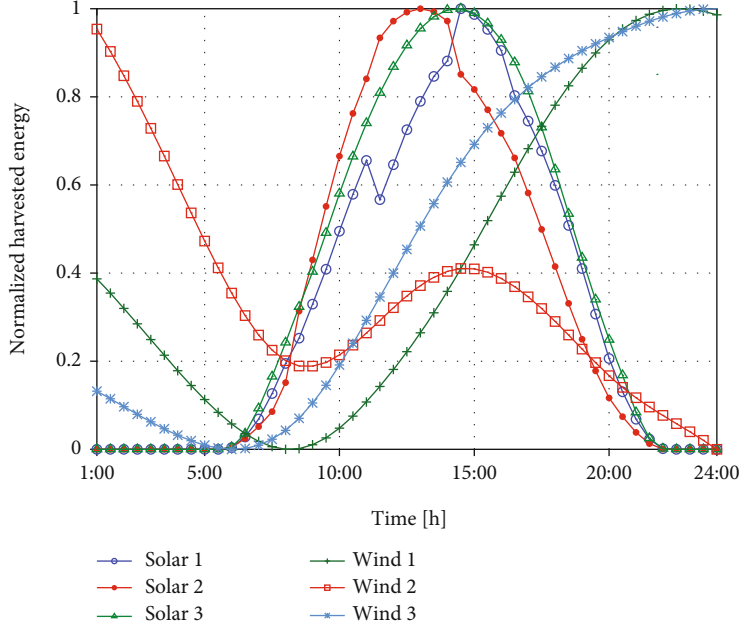
FIGURE 3: Example traces for harvested solar traces and wind traces from [43].

For example, when a video becomes viral, users watching it share and talk about that video, which will be then requested by other users after a response time. Taking into account the Internet users' response time $\bar{\lambda}(t)$, this epidemic behavior can be modelled by the self-excited Hawkes condition Poisson process described in [42]: $\bar{\lambda}(t) = V(t) + \sum_{t_i \leq t} \Omega_i k_i(t - t_i)$, where $k_i(t)$ is the response time function and $\Omega_i$ is the number of potential viewers who will be influenced after $t_i$, which is the time when the user $i$ shared the video. The term $V(t)$ is added as a component of the model to capture the views that are not triggered by the epidemic effect. The energy consumption is mainly contributed by content caching and data transmission processes, as such $\theta_{CC}(t)$ is defined as:

$$\theta_{CC}(t) = \bar{\lambda}(t) \left( \theta_{TR}(t) + \theta_{CACHE}(t) \right), \tag{9}$$

where $\theta_{TR}(t)$ is the power consumption due to transmission and $\theta_{CACHE}(t)$ is the power consumption contributed by the caching process.

*3.2. Energy Patterns and Storage.* The rechargebale energy storage device is characterized by its finite energy storage capacity $b_{max}$. At each time instance, the energy level reports are pushed from the BS sites to the MEC server. Thus, the EB level $b(t)$ is known, enabling the provisioning of the required computation and communication resources, i.e., the required containers, transmission drivers, and BSs to be active. In this paper, the amount of harvested energy $H(t)$, per site BS site, in time slot $t$ is obtained from open-source solar and wind traces from a farm located in Belgium [43] (see Figure 3 above). The data in the dataset matches our time slot duration (30). The dataset is the result of daily environmental records for a place assumed to be free from surrounding obstructions (e.g., buildings, shades).

The harvested energy $H(t)$ is obtained by picking a day at random in the dataset and associating it with one site. Here, the wind energy is selected as a source during the solar energy off-peak periods. The available EB level $b(t + 1)$ located at the BS site (BS $n$) or computing platform evolves according to the following dynamics:

$$b(t + 1) = \min \{ b(t) + H(t) - \theta_{site}(t) - a(t) + E(t), b_{max} \}, \tag{10}$$

where $b(t)$ is the energy level in the battery at the beginning of the time slot $t$ and $\theta_{site}(t)$ represents either $\theta_{BS,n}(t)$, the BS energy consumption of the communication site, or $\theta_{MEC}(t)$, the energy drained at the computing platform, over time slot $t$, see Eqs. (2) and (3). $a(t)$ is leakage energy, and $E(t) \geq 0$ is the amount of energy purchased from the power grid. It is worth noting that $b(t)$ is updated at the beginning of time slot $t$, whereas $H(t)$, $\theta_{BS,n}(t)$, and $\theta_{MEC}(t)$ are only known at the end of it. Thus, the energy constraint at the computing site must be satisfied for every time slot: $\theta_{MEC}(t) \leq b(t)$.

For decision-making in the GENM application, the received EB level reports are compared with the following thresholds: $b_{low}$ and $b_{up}$, respectively, termed the lower and the upper energy threshold with $0 < b_{low} < b_{up} < b_{max}$. $b_{up}$ corresponds to the desired energy buffer level at the BS site or computing site, and $b_{low}$ is the lowest EB level that any site should ever reach. If $b(t) < b_{low}$, then BS $n$ or the computing site is said to be *energy deficient*. The suitable energy source at each time slot $t$ is selected based on the forecast expectations, i.e., the expected harvested energy $\hat{H}(t)$. If $\hat{H}(t)$ is enough to reach $b_{up}$, no energy purchase is needed. Otherwise, the remaining amount up to $b_{up}$, i.e., $E(t) = b_{up} - b(t)$, is purchased from the electrical grid. Our optimization framework

in Section 4.1 makes sure that $b(t)$ never falls below $b_{\text{low}}$ and guarantees that $b_{\text{up}}$ is reached at every time slot.

# 4. Problem Formulation

In this section, the optimization problem is formulated to obtain reduced energy consumption through short-term traffic load and harvested solar energy forecasting along with energy management procedures. The optimization problem is defined in Section 4.1, and the edge system management procedures are presented in Section 4.2.

*4.1. Optimization Problem.* Our objective is to improve the overall energy savings of the edge system through BS power saving modes (i.e., green-based traffic load balancing), autoscaling of containers, contents caching, and tuning of the transmission drivers, and also to guarantee the QoS within the network. Note that at the end of each time slot, the EB states are updated depending on the harvested energy and the consumed energy, thereby linking per-time slot problems across time.

To achieve our objective, two cost functions are defined, one captures the edge system energy consumption and the other handles the QoS. This is defined as (F1)$\theta_{\text{EDGE}}(t)$, which weighs the energy consumption due to transmission in the BSs and the computing-plus-communication activities in the MEC server and (F2) a quadratic term$(\xi(t) - L_{\text{in}}(t))^2$, which accounts for the QoS. In this regard, it is worth noting that F1 tends to push the system towards self-sustainability solutions and F2 favors solutions where the delay-sensitive load is entirely admitted in the MEC server by the router application, taking into account the expected energy to be harvested in the computing site. A weight $\Gamma = [0, 1]$ is utilized to balance the two objectives F1 and F2. The corresponding (weighted) cost function is defined as:

$$J(\delta, \psi, M, t) \stackrel{\Delta}{=} \bar{\Gamma}\theta_{\text{EDGE}}(\delta_n(t), \{\psi_c(t)\}, M(t), t) + \Gamma(\xi(t) - L_{\text{in}}(t))^2,$$
(11)

where $\bar{\Gamma} \stackrel{\Delta}{=} 1 - \Gamma$. Hence, starting from $t = 1$ (i.e., $t = 1, 2, \cdots, T$) as the current time slot and the finite horizon $T$, the following optimization problem is formulated as:

$$P1 : \min_{\mathscr{E}} \sum_{t=1}^{T} J(\delta, \psi, M, t),$$
(12)

subject to:
A1: $\delta_n(t) \in \{\varepsilon, 1\}$,
A2: $\beta \le C(t) \le C$,
A3: $b(t) \ge b_{\text{low}}$,
A4: $0 \le f_c(t) \le f_{\text{max}}$,
A5: $0 \le \lambda_c(t) \le \lambda_{\text{max}}$,
A6: $\chi_c(t) \le \Delta$,
A7: $\sum_{c=1}^{C(t)} r_c(t) \le r_{\text{max}}$,
A8: $\theta_{\text{MEC}}(t) \le b(t)$,
A9: $\max\{2\rho_c(t)\} + \Delta = \tau_{\text{max}}, t = 1, \cdots, T$,

where the set of objective variables to be configured at slot $t$ in the BS system and MEC server is defined as

$\mathscr{E} = {}^{\Delta}\{\{\delta_n(t)\}, C(t), \{\psi_c(t)\}, \{P_c(t)\}, \{\lambda_c(t)\}, \zeta(t), M(t)\}$. The setting handles the transmission and computing-plus-communication activities. Constraint A1 specifies the BS operation status (either *power saving* or *active*), and A2 forces the required number of containers, $C(t)$, to be always greater than or equal to a minimum number $\beta \ge 1$: the purpose of this is to be always able to handle mission-critical communications. A3 makes sure that the EB level is always above or equal to a preset threshold $\beta_{\text{low}}$, to guarantee *energy self-sustainability* over time. Furthermore, A4 and A5 bound the maximum processing rate and workloads of each running container $c$, with $c = 1, \cdots, C(t)$, respectively. Constraint A6 represents a hard-limit on the corresponding per-slot and per-VM processing time. A7 bounds the aggregate communication rate sustainable by the VLAN to $r_{\text{max}}$, and A8 ensures that the energy consumption at the computing site (due to the admitted computational workload) is bounded by the available energy in the EB. A9 forces the server to process the offloaded tasks within the set value $\tau_{\text{max}}$.

From the optimization problem P1, it could be noted that $J(\zeta, \psi, M, t)$ consists of a nonconvex component defined in Eq. (7), while the others are convex and nondecreasing. In this case, Eq. (7) can be converted into a convex function using the GP concept [44], by introducing alternative variables and approximations. In this case, fixed parameters and approximations are introduced, i.e., $\mu_c$, $\nu_c$. In the sequel, the index $t$ is dropped to improve readability. Thus, letting $r_c = 2 \lambda_c/(\tau_{\text{max}} - \Delta)$ and then obtaining $P_c(r_c)$ in terms of $\lambda_c$ by rearranging the Shannon-Hartley expression and substituting the value of $r_c$: $\widehat{P}_c(r_c) = (((2\lambda_c/(\tau_{\text{max}} - \Delta)) - \nu_c W_c) \ln 2)/(\mu_c W_c) + \ln (N_0^{(c)}) - \ln g_c$. From the Shannon-Hartley expression, the presence of the log-sum-exp function is observed as it has been proven to be convex in [45] and recall that $P_c(r_c) = \exp(\widehat{P}_c(r_c))$.

To solve P1 in (12), the LLC principles [28, 31], GP technique [44], and heuristics is used towards obtaining the feasible system control inputs $\varphi(t) = (\{\delta_n(t)\}, C(t), \{\psi_c(t)\}, \{P_c(t)\}, \{\lambda_c(t)\}, \zeta(t), M(t))$ for $t = 1, \cdots, T$. Note that (12) can iteratively be solved at any time slot $t \ge 1$, by just redefining the time horizon as $t' = t, t + 1, \cdots, t + T - 1$.

*4.2. Edge System Management.* In this subsection, a traffic load and energy harvesting prediction method and an online management algorithm are proposed to solve the previously stated problem P1.

*4.2.1. Traffic Load and Energy Prediction.* Given a time slot duration of $\tau = 30$, the time series prediction is performed, i.e., the $T = 3$ estimates of $\widehat{L}_n(t)$ and $\widehat{H}(t)$ are obtained by using an LSTM developed in Python using Keras deep learning libraries (Sequential, Dense) where the network has a one-dimensional (1D) subsequence of data, single feature, and multistep for an output. The dataset is split as 70% for training and 30% for testing. The efficient Adam implementation of stochastic gradient descent and fit the model for 20 epochs with a batch size of 4 is used. As for the performance measure of the model, the RMSE is used.

*4.2.2. Edge System Dynamics.* The system state vector at time $t$ is denoted by $q(t) = (\delta(t), C(t), M(t), b(t))$, which contains the number of active BS, $\delta(t)$; number of active containers, $C(t)$; transmission drivers for data transfers, $M(t)$; and the EB level, $b(t)$. The input vector $\varphi(t) = (\{\delta_n(t)\}, C(t), \{\psi_c(t)\}, \{P_c(t)\}, \{\lambda_c(t)\}, \zeta(t), M(t))$ drives the MEC server behavior (handles the joint switching on/off of BSs, autoscaling and reconfiguration of containers, and the tuning of transmission drivers) at time $t$. In this work, $\{P_c^*(t)\}$ is obtained with CVXOPT toolbox, and $\{\lambda_c^*(t)\}$ is obtained by following the procedure outlined in Remark 1 (M. Andersen and J. Dahl. CVXOPT: Python Software for Convex Programming, 2019. [Online]. Available: https://cvxopt.org/).

The system behavior is described by the discrete-time state-space equation, adopting the LLC principles [14, 28]:

$$q(t+1) = \phi(q(t), \varphi(t)), \tag{13}$$

where $\phi(\cdot)$ is a behavioral model that captures the relationship between $(q(t), \varphi(t))$ and the next state $q(t+1)$. Note that this relationship accounts for the amount of energy drained $\theta_{\text{COMM}}(t), \theta_{\text{MEC}}(t)$ that harvested $H(t)$ and that purchased from the electrical grid $E(t)$, which together lead to the next buffer level $\beta(t+1)$ through Eq. (10). The GENM algorithm finds the best control action vector that yields the desired energy savings within the edge network. Specifically, for each time slot $t$, problem (12) is solved, obtaining control actions for the prediction horizon $T$. The control action that is applied at time $t$ is $\varphi^*(t)$, which is the first one in the retrieved control sequence. This control amounts to setting the number of active BSs, $\{\delta_n^*(t)\}$; number of instantiated containers, $C^*(t)$ (along with their obtained $\{\psi_c^*(t)\}$, $\{P_c^*(t)\}$, $\{\lambda_c^*(t)\}$ values); NIC status to either active or not, $\zeta^*(t) \in (0, 1)$; and the optimal transmission drivers, $M^*(t)$. The entire process is repeated every time slot $t$ when the controller can adjust the behavior given the new state information.

State $q(t)$ and $\varphi(t)$ are, respectively, measured and applied at the beginning of the time slot $t$, whereas the offered load $L(t)$ and the harvested energy $H(t)$ are accumulated during the time slot and their value becomes known only by the end of it. This means that, being at the beginning of the time slot $t$, the system state at the next time slot $t+1$ can only be estimated, which is formally written as:

$$\hat{q}(t+1) = \phi(q(t), \varphi(t)). \tag{14}$$

*Remark 1* (container provisioning and load distribution). For a fair provisioning of the computing resources, $C(t)$, and the expected workload allocation, $\hat{\xi}(t+1)$, a remark is presented. Firstly, each container can only compute an amount of up to $\lambda_{\max}$, and to meet the latency requirements, $C(t)$ is obtained as $C(t) = \lceil (\hat{\xi}(t+1)/\lambda_{\max}) \rceil$, where $\lceil \bullet \rceil$ returns the nearest upper integer. Secondly, to distribute the workload among the $C(t)$ containers, a heuristic process splits the computational workload $\lambda_c(t) = \lambda_{\max}$ to the first $C(t) - 1$ containers, and the remaining workload $\lambda_c(t) = \hat{\xi}(t+1) - (C(t) - 1)\lambda_{\max}$ to the last one.

*4.2.3. Edge System Management Framework.* In order to perform traffic load balancing using the green energy as a performance metric, a framework is defined that will identify the BS to be dynamically switched off and then steer the traffic load towards those BS with sufficient green energy. To do this, the available operating interval is defined as the ratio of the next time slot available green energy and the expected total power consumption (recall that the BS load is forecasted), per BS site, as

$$I_n(t) = \frac{b_n(t+1)}{\theta_{\text{BS},n}(t+1)} \geq 1. \tag{15}$$

If $I_n(t) < 1$, the BS site will not have sufficient energy to handle the expected traffic and it becomes a potential BS to be switched off. In the case where $I_n(t) \geq 1$, the site energy will be sufficient to handle the expected traffic. The potential BS to be switched off, denoted by BS $n$, will offload its traffic load to a neighboring BS, denoted by BS $nn'$. For BS $nn'$ to be able to handle the offloaded traffic, the energy must be sufficient; thus, the green-based operating interval is defined as

$$x_{nn'}(t) = \frac{b_{nn'}(t+1)}{\theta_{\text{BS},nn'}(t+1)}, \tag{16}$$

where $\theta_{\text{BS},nn'}(t+1)$ is the total energy consumption of the BS site when the traffic load from the neighboring BS is combined with the expected load of the BS, and $b_{nn'}(t+1)$ is the next time slot energy. Next, the BS wake-up procedure is discussed.

*(1) BS Wake-Up Procedure.* To support BS reactivation commands, the UE location fingerprints that are obtained from the LS API are considered. The UE trajectory is assumed to be sequential, i.e., from BS-to-BS along the same direction (while still associated with the MEC server), and this is represented as $i_1(t) \rightarrow i_2(t) \rightarrow \cdots \rightarrow i_n(t)$, where $i_n(t)$ refers to the serving/target BS node $n$ in association with user $i$, at time slot $t$. When a BS node is switched off, it goes into a discontinuous reception cycle and configures a timer to awake and listen. Here, the MEC server, as BSs manager, sends wake-up control information as wake-up signaling (the information is a single bit). The wake-up information is only sent during the listening period. A BS $n$ can be woken up only if it meets the following conditions: (i) $b_n(t+1) > b_{\text{low}}$ and (ii) a group of UEs that are associated with the MEC server is expected to receive the computed results via BS $n$ (their trajectory is towards BS $n$ as reported by the LS in the MEC).

*4.2.4. Green-Based Edge Network Management (GENM) Algorithm.* In order for the algorithm to manage the BS system, deciding upon the allocation of their transmission resources and also handling the computing and communication process, the best control action, $\varphi(t) = (\{\delta_n(t)\}, C(t), \{\psi_c(t)\}, \{P_c(t)\}, \{\lambda_c(t)\}, \zeta(t), M(t))$, that will yield the expected system behavior shall be obtained.

```
Input: q(t) (current state)
Output: φ*(t) (control input vector)
01: Parameter initialization
     𝒮(t) = {q(t)}.
02: for (k within the prediction horizon of depth T) do
     - L̂_in(t + k):= forecast the workload
     - Ĥ_n(t + k):= forecast the energy
     - I_n(t + k):= operating interval of each BS
     - x_nn'(t + k):= green-based operating interval
     - 𝒮(t + k) = ∅
03:  for (each q(t) in 𝒮(t + k)) do
     - generate all reachable states q̂(t + k)
     - 𝒮(t + k) = 𝒮(t + k) ∪ {q̂(t + k)}
04:  for (each q̂(t + k) in 𝒮(t + k)) do
     - calculate the corresponding θ_EDGE(q̂(t + k))
     taking into account of κ_n from L_out(t)
     end for
        end for
           end for
05: - obtain a sequence of reachable states yielding
     minimum energy cost
06: φ*(t) := control leading from q(t) to q̂_min
07: Return φ*(t)
```

ALGORITHM 1. GENM algorithm pseudocode.

TABLE 2: System parameters.

| Parameter | Value |
| --- | --- |
| Max. number of containers, $C$ | 20 |
| Min. number of containers, $\beta$ | 1 |
| Time slot duration, $\tau$ | 30 |
| Idle state energy for container $c$, $\theta_{\mathrm{idle}_c}(t)$ | 4 |
| Max. energy for container $c$, $\theta_{\max,m}(t)$ | 10 |
| Per-container reconfiguration cost, $z_e$ | 0.005 J/(MHz)$^2$ |
| TOE in idle state, $\theta_{\mathrm{idle}}^{\mathrm{NIC}}(t)$ | 13.1 J |
| Max. allowed processing time, $\Delta$ | 0.8 |
| Processing rate set, $\{f_c(t)\}$ | {0,50,70,90,105} |
| Bandwidth, $W_c$ | 1 MHz |
| Max. number of drivers, $M$ | 6 |
| Noise spectral density, $N_0^{(c)}$ | -174 dBm/Hz |
| Max. container $c$ load, $\lambda_{\max}$ | 10 MB |
| NIC best performance throughput, $\eta$ | 1.4 Gbit/J |
| Driver energy, $d_m(t)$ | 1 J/s |
| Target transmission rate, $r_0$ | 1 Mbps |
| Controllable factor of delay, $u$ | 0.96 |
| Reconfiguration overhead, $\sigma$ | 20 ms |
| Leakage energy, $a(t)$ | 2 $\mu$J |
| Energy storage capacity, $b_{\max}$ | 490 |
| Lower energy threshold, $b_{\mathrm{low}}$ | 30% of $b_{max}$ |
| Upper energy threshold, $b_{\mathrm{up}}$ | 70% of $b_{max}$ |

The edge network management algorithm pseudocode is outlined in Algorithm 1, and it is based on the LLC principles from [14, 28]. Starting from the *initial state*, the controller constructs, in a breadth-first fashion, a tree comprising all possible future states up to the prediction depth $T$. The algorithm proceeds as follows: a search set $\mathcal{S}$ consisting of the current system state is initialized (line 01), and it is accumulated as the algorithm traverse through the tree (line 03), accounting for predictions, accumulated workloads at the output buffer, mobile devices trajectory $i_n(t)$, past outputs and controls, and operating intervals. The set of states reached at every prediction depth $t + k$ is referred to as $\mathcal{S}(t + k)$ (line 02). Given $q(t)$, the workload $\hat{L}_{\mathrm{in}}(t + k)$ and harvested energy $\hat{H}(t + k)$ are estimated first, then obtain the operating intervals $I_n(t + k), x_{nn'}(t + k)$ (line 02), and generate the next set of reachable control actions by applying the accepted workload $\xi(t + k)$, energy harvested, and green-based operating interval (line 03). The energy cost function corresponding to each generated state $\hat{q}(t + k)$ is then computed (line 04), where $\hat{q}(t + k)$ take into account of $\eta_n$ as observed from $L_{\mathrm{out}}(t)$. Once the prediction horizon is explored, a sequence of reachable states yielding minimum energy consumption is obtained (line 05). The control action $\varphi^*(t)$ corresponding to $\hat{q}(t + k)$ (the first state in this sequence) is provided as input to the system while the rest are discarded (line 06). The process is repeated at the beginning of each time slot $t$.

*(1) Algorithm Complexity.* The algorithm is executed at each time instance, and the corresponding time complexity is obtained as follows. The time complexity associated with the computation of the $I_n(t)$ and $x_{nn'}$ is linear with the size of the BS group $|N|$ interconnected to the MEC server. Next, the complexity associated with updating the load allocation for the active BSs is $|N| - 1$, which leads to $O(|N|^2)$. In the worst-case scenario (no BS has been switched off), the total complexity is $|N| q(t)\varphi(t)T$, which is linear in all variables, namely, number of BSs interconnected to the MEC server, number of system states, number of control actions, and time horizon $T$.

## 5. Performance Evaluation

In this section, some selected numerical results for the scenario of Section 3 are shown. The parameters that were used in the simulations are listed in Table 2.

*5.1. Simulation Setup.* A virtualized MEC server in proximity to a group of BS is considered. The BS coverage areas overlap to enable load balancing. Our time slot duration $\tau$ is set to 30, and the time horizon is set to $T = 3$ time slots. For simulation, Python is used as the programming language.

*5.2. Numerical Results. Data preparation*: The information from the used mobile and energy traces is aggregated to the set time slot duration. The mobile traces are aggregated from 10 observation time to $\tau$. As for the wind and solar traces,
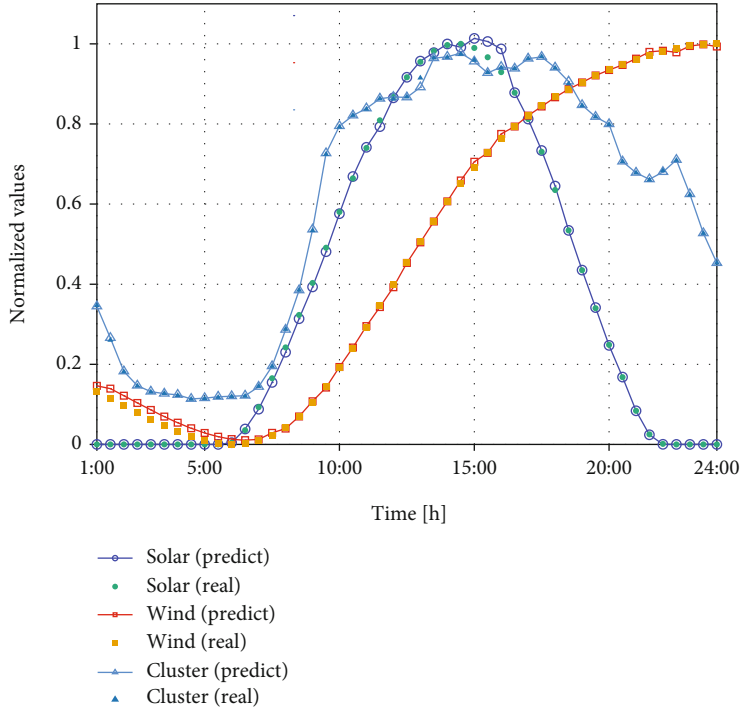
FIGURE 4: One-step ahead predictive values for $L(t)$ and $H(t)$.

they were aggregated from 15 observation time to $\tau$. The used datasets are readily available in a public repository (see [39]).

In Figure 4, the real and predicted values for BS traffic load and harvested energy is shown. Here, the forecasting routing tracks each value and predicts it over one-step. The shown selected prediction results are for Cluster 3, Solar 3, and Wind 3. Then, Table 3 shows the average RMSE of the normalized harvested energy and traffic load processes, for different time horizon values, $T \in \{1, 2, 3\}$. In the table, the term $H_{\text{wind}}(t)$ represents the forecasted values for energy harvested from wind turbines, and $H_{\text{solar}}(t)$ is for the harvested energy from solar panels. From the obtained results, the prediction variations are observed between $H(t)$ and $L(t)$ when comparing the average RMSE. The measured accuracy is deemed good enough for the proposed optimization.

The GENM algorithm is benchmarked with another one, named Iterative-based Resource Manager with network impact Capability (IRMC), which is inspired by the iterative approach for computing platforms from [20] and the use of the network impact towards load balancing from [5]. Both algorithms make use of the learned information. Figures 5 and 6 show the average energy savings obtained by GENM in the MEC server. In Figure 5(a), the average results for GENM $(z_e = 0.005, |N| = 24, \Gamma = 0.5, \lambda_{\text{max}} = 10 \text{ MB})$ show energy savings of 59%, while IRMC achieves 34% on average. As expected, the highest energy savings gain is observed in the early hours of the day (1-8) as the aggregated computational workload was at its lowest. In Figure 5(b), the average energy savings obtained by GENM is 68% $(z_e = 0.005, |N| = 12, \Gamma = 0.5, \lambda_{\text{max}} = 10 \text{ MB})$ and for IRMC is 49%. Again, here, the highest peaks for energy savings are obtained from 1-8. The results are obtained with respect to the case where no

TABLE 3: Average prediction error (RMSE) for harvested energy and traffic load processes, both normalized in [0,1].

|                      | $T = 1$ | $T = 2$ | $T = 3$ |
|----------------------|---------|---------|---------|
| $L(t)$               | 0.010   | 0.013   | 0.018   |
| $H_{\text{wind}}(t)$ | 0.011   | 0.013   | 0.016   |
| $H_{\text{solar}}(t)$| 0.010   | 0.011   | 0.014   |

energy management procedures are applied, i.e., the MEC server provisions the computing resources for maximum expected computation workload (maximum value of $\theta_{\text{MEC}}(t)$, $C = 20, \forall t$). Comparing the results of Figures 5(a) and 5(b), we observed that when the BSs being manage by the MEC server are reduced (i.e., $12 < 24$), the aggregated delay-sensitive workload is also reduced and this translates to reduced computation process demands, which in turn results into high energy savings.

Figure 6 shows the average energy savings obtained when green energy is used as a performance metric towards load balancing within a group of BSs. Here, the group size is increased from $|N| = 5$ to 50, using an incremental step size of 5. The obtained energy savings are with respect to the case where all BSs are dimensioned for maximum expected capacity (maximum value of $\theta_{\text{COMM}}(t)$). From the results, it is observed that the energy savings increase as the BS cluster grows, thanks to the load balancing among active BSs.

Then, Figure 7 shows the average energy savings for the edge system. Here, the BS group size is set to $|N| = 40$, and the obtained energy saving results are with respect to the case where no energy management procedures are applied, i.e., the BSs are dimensioned for maximum expected capacity
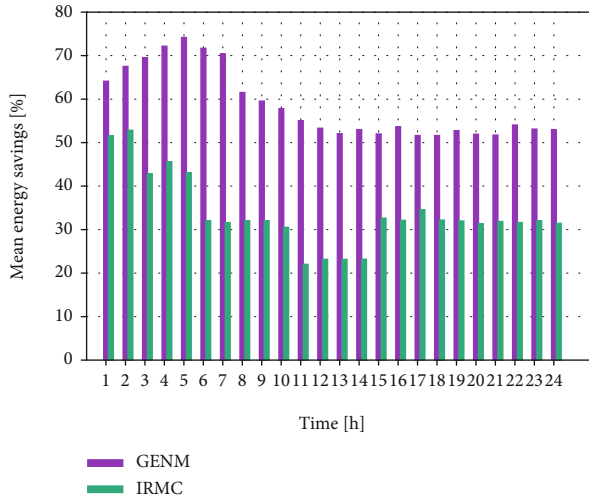
(a) Mean energy savings for $\Gamma = 0.5$, $|N| = 24$, $\lambda_{\max} = 10$ MB
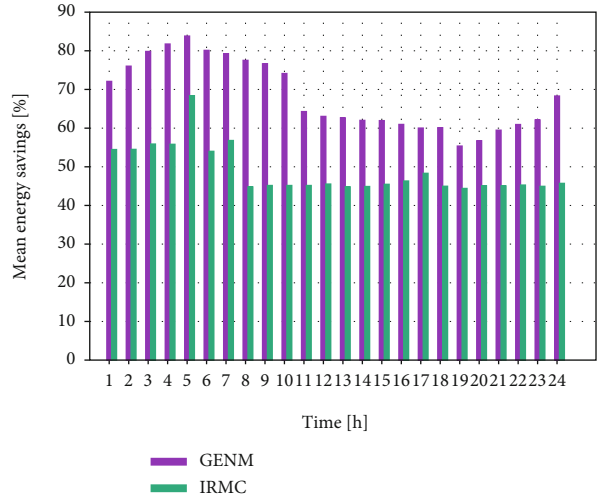
(b) Mean energy savings for $\Gamma = 0.5$, $|N| = 12$, $\lambda_{\max} = 10$ MB

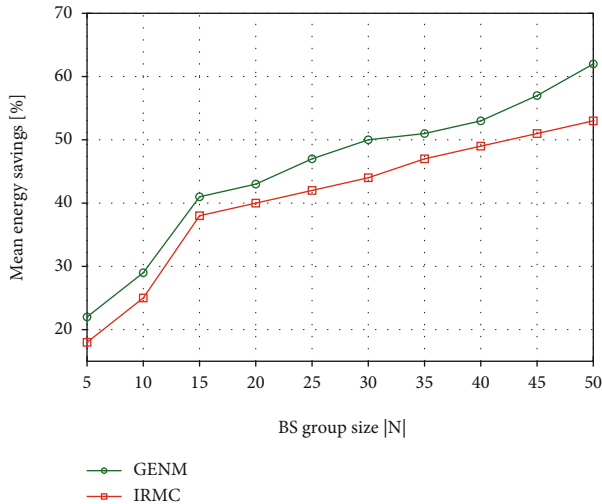FIGURE 5: Mean energy savings within the MEC server.



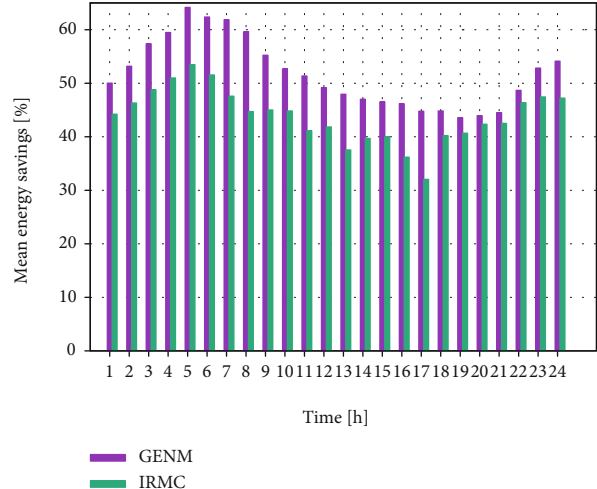FIGURE 6: Energy savings versus BS group size.



FIGURE 7: Mean energy savings for the edge system.

(maximum value of $\theta_{\mathrm{COMM}}(t)$, $\forall t$) and the MEC server provisions the computing resources for the maximum expected computation workload (maximum value of $\theta_{\mathrm{MEC}}(t)$, with $C = 20$ containers, $\forall t$). The average results of GENM ($z_e = 0.05$, $\lambda_{\max} = 10$ MB, $\Gamma = 0.5$) show energy savings of 51%, while IRMC achieves 44% on average. The effectiveness of the joint dynamic BS management, autoscaling and reconfiguration of the computing resources, and on/off switching of the fast tunable laser drivers, coupled with foresighted optimization, is observed in the obtained numerical results.

## 6. Conclusions

This paper envisioned an edge system where a group of BS is placed in proximity to a MEC server for ease of handling the offloaded computational workload and BS management, and also, the edge apparatuses are powered by hybrid supplies, i.e., green energy is used in order to promote energy self-sustainability within the network and as a performance metric for traffic load balancing. The extra energy can only be purchased from the grid supply to supplement the renewable energy supplies. The considered energy cost model takes into account the computing, caching, and communication processes within the MEC server, and the transmission-related energy consumption in BSs. To intelligently manage the edge system, an online algorithm based on forecasting, control theory, and heuristics is proposed with the goal of minimizing the overall energy consumption and to guarantee the quality of service within the network. The algorithm jointly performs (i) dynamic BS management using green energy as a performance metric; (ii) autoscaling and reconfiguration of the computing resources, workload, and processing rate allocation; and lastly, (iii) switching on/off of fast tunable drivers. Numerical results, obtained with real-world energy and traffic load traces, demonstrate that the proposed algorithm achieves energy savings of above 50% with respect to

the allocated maximum per-container loads of 10 MB. The computing platform is able to achieve energy savings from 59% to 68%, depending on the size of the BS cluster. The energy-saving results are obtained with respect to the case where no energy management techniques are applied in the BS system and the MEC server.

## Data Availability

In this paper, open-source datasets for the mobile network (MN) traffic load and solar and wind energy have been used. The details are as follows: (1) the real MN traffic load traces used to support the findings of this study were obtained from the Big Data Challenge organized by Telecom Italia Mobile (TIM) and the data repository has been cited in this article. (2) The real solar and wind traces used to support the findings of this study have also been cited in this article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, 2018.

[2] "Software-Defined and Cloud-Native Foundations for 5G Networks," https://www.interdigital.com/all_white_papers.

[3] T. Han and N. Ansari, "A traffic load balancing framework for software-defined radio access networks powered by hybrid energy sources," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 1038–1051, 2016.

[4] J. Xu, H. Wu, L. Chen, C. Shen, and W. Wen, *Online Geographical Load Balancing for Mobile Edge Computing with Energy Harvesting*, IEEE International Conference on Communications (ICC), Kansas, USA, 2018.

[5] E. Oh, K. Son, and B. Krishnamachari, "Dynamic base station switching-on/off strategies for green cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 5, pp. 2126–2136, 2013.

[6] T. Dlamini, Á. F. Gambín, D. Munaretto, and M. Rossi, "Online supervisory control and resource management for energy harvesting BS sites empowered with computation capabilities," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 8593808, 17 pages, 2019.

[7] R. Morabito, "Power Consumption of Virtualization Technologies: An Empirical Investigation," in *IEEE International Conference on Utility and Cloud Computing (UCC)*, pp. 522–527, Limassol, Cyprus, 2015.

[8] Y. Jin, Y. Wen, and Q. Chen, "Energy Efficiency and Server Virtualization in Data Centers: An Empirical Investigation," in *2012 Proceedings IEEE INFOCOM Workshops*, pp. 133–138, Orlando, USA, 2012.

[9] T. Dlamini, "Softwarization in future mobile networks and energy efficient networks," *Mobile Computing*, 2019, https://www.intechopen.com/online-first/softwarization-in-future-mobile-networks-and-energy-efficient-networks=0pt.

[10] S. Fu, H. Wen, J. Wu, and B. Wu, "Cross-networks energy efficiency tradeoff: from wired networks to wireless networks," *IEEE Access*, vol. 5, pp. 15–26, 2017.

[11] T. Dlamini and A. F. Gambin, "Adaptive resource management for a virtualized computing platform within edge computing," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, Boston, USA, 2019.

[12] M. Portnoy, *Virtualization essentials*, John Wiley and Sons, 2012.

[13] E. Oh, B. Krishnamachari, X. Liu, and Z. Niu, "Toward dynamic energy-efficient operation of cellular network infrastructure," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 56–61, 2011.

[14] S. Abdelwahed, N. Kandasamy, and S. Neema, "Online Control for Self-Management in Computing Systems," in *Proceedings. RTAS 2004. 10th IEEE Real-Time and Embedded Technology and Applications Symposium, 2004*, pp. 368–375, Ontario, Canada, 2004.

[15] A. Bousia, E. Kartsakli, A. Antonopoulos, L. Alonso, and C. Verikoukis, "Multiobjective auction-based switching-off scheme in heterogeneous networks: to bid or not to bid?," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 11, pp. 9168–9180, 2016.

[16] T. Han and N. Ansari, "On optimizing green energy utilization for cellular networks with hybrid energy supplies," *IEEE Transactions on Wireless Communications*, vol. 12, no. 8, pp. 3872–3882, 2013.

[17] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[18] X. Jie and R. Shaolei, "Online learning for offloading and auto-scaling in renewable-powered mobile edge computing," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, USA, 2016.

[19] T. Dlamini, Á. F. Gambn, D. Munaretto, and M. Rossi, "Online resource management in energy harvesting BS sites through prediction and soft-scaling of computing resources," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1820–1826, Bologna, Italy, 2018.

[20] M. Shojafar, N. Cordeschi, D. Amendola, and E. Baccarelli, "Energy-saving adaptive computing and traffic engineering for real-time-service data centers," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, pp. 1800–1806, London, UK, 2015.

[21] M. Shojafar, N. Cordeschi, and E. Baccarelli, "Energy-efficient adaptive resource management for real-time vehicular cloud services," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 196–209, 2019.

[22] M. Mukherjee, V. Kumar, S. Kumar et al., "Computation offloading strategy in heterogeneous fog computing with energy and delay constraints," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–5, Dublin, Ireland, 2020.

[23] J. Bushra, S. Mohammad, A. Israr, U. Atta, M. Kashif, and I. Humaira, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, 2020.

[24] M. Mithun, K. Suman, S. Mohammad, Z. Qi, and X. Mavromoustakis Constandinos, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, Shanghai, China,, 2019.

[25] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, Paris, France,, 2017.

[26] B. Wu, S. Fu, X. Jiang, and H. Wen, "Joint scheduling and routing for QoS guaranteed packet transmission in energy efficient reconfigurable WDM mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 8, pp. 1533–1541, 2014.

[27] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *2008 International Conference on Autonomic Computing*, pp. 3–12, Chicago, USA, 2008.

[28] N. Kandasamy, S. Abdelwahed, and J. P. Hayes, "Self-optimization in computer systems via on-line control: application to power management," in *International Conference on Autonomic Computing, 2004. Proceedings*, pp. 54–61, Washington, USA, 2004.

[29] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, OTexts, Melbourne, Australia, 2013.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[31] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1921–1935, 1992.

[32] A. Ferdowsi, U. Challita, and W. Saad, "Deep Learning for Reliable Mobile Edge Analytics in Intelligent Transportation Systems: An Overview," *IEEE Vehicular Technology Magazine*, vol. 14, pp. 62–70, 2019.

[33] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018.

[34] S. Kekki, W. Featherstone, Y. Fang et al., *MEC in 5G Networks*, ETSI white paper, Sophia-Antipolis, France, 2018.

[35] S. Ripduman, R. Andrew, A. W. Moore, and M. Kieran, "Characterizing 10 Gbps network interface energy consumption," in *IEEE Local Computer Network Conference*, pp. 268–271, Colorado, USA, 2010.

[36] "Open big data challenge," https://dandelion.eu/datamine/open-big-data/.

[37] D. Pelleg and A. W. Moore, "X-means: extending k-means with efficient estimation of the number of clusters," in *Proceedings of the 17th International Conf. on Machine Learning*, San Francisco, USA, 2000.

[38] L. Chen, S. Zhou, and J. Xu, "Energy efficient mobile edge computing in dense cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, Paris, France, 2017.

[39] T. Dlamini, "Mobile and energy datasets," https://github.com/lihles/mobile-datasets=0pt.

[40] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*, pp. 327–334, New York, USA, 2009.

[41] N. Cordeschi, M. Shojafar, and E. Baccarelli, "Energy-saving self-configuring networked data centers," *Computer Networks*, vol. 57, no. 17, pp. 3479–3491, 2013.

[42] F. B. Abdesslem and A. Lindgren, "Large scale characterisation of YouTube requests in a cellular network," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pp. 1–9, Sydney, Australia, 2014.

[43] "Total generation," https://www.elia.be/en/grid-data/power-generation.

[44] W.-C. Ho, L.-P. Tung, T.-S. Chang, and K.-T. Feng, "Enhanced component carrier selection and power allocation in LTE-advanced downlink systems," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 574–579, Shanghai, China, 2013.

[45] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.