Journal of Network and Systems Management (2024) 32:53
https://doi.org/10.1007/s10922-024-09825-9

Energy-Aware Microservice-Based Application Deployment in UAV-Based Networks for Rural Scenarios

Diego Ramos-Ramos[1] · Alejandro González-Vegas[1] · Javier Berrocal[1] · Jaime Galán-Jiménez[1]

Received: 31 May 2023 / Revised: 9 April 2024 / Accepted: 13 April 2024 /
Published online: 31 May 2024

Abstract

Yearly, the rates of Internet penetration are on the rise, surpassing 80% in developed nations. Despite this progress, over two billion individuals in rural and low-income regions face a complete absence of Internet access. This lack of connectivity hinders the implementation of vital services like remote healthcare, emergency assistance, distance learning, and personal communications. To bridge this gap and bring essential services to rural populations, this paper leverages Unmanned Aerial Vehicles (UAVs). The proposal introduces a UAV-based network architecture and an energy-efficient algorithm to deploy Internet of Things (IoT) applications. These applications are broken down into microservices, strategically distributed among a subset of UAVs. This approach addresses the limitations associated with running an entire IoT application on a single UAV, which could lead to suboptimal outcomes due to battery and computational constraints. Simulation results conducted in a realistic scenario underscore the effectiveness of the proposed solution. The evaluation includes assessing the percentage of IoT requests successfully served to users in the designated area and reducing the energy consumption required by UAVs during the handling of such requests.

Keywords Energy efficiency · UAV · IoT · Microservices · Digital divide

✉ Diego Ramos-Ramos
diegorr@unex.es

Alejandro González-Vegas
agonzalewc@alumnos.unex.es

Javier Berrocal
jberolm@unex.es

Jaime Galán-Jiménez
jaime@unex.es

[1] Departamento de Ingeniería de Sistemas Informáticos y Telemáticos, Universidad de Extremadura, Av. de la Universidad, 10003 Cáceres, Extremadura, Spain

1 Introduction

The profound influence of technology on the world has brought about significant transformations in various spheres. Businesses have witnessed enhanced efficiency, with seamless communication facilitated through video conferencing, connecting companies across different locations. The rapid processing of information has enabled real-time services in remote e-Health, entertainment, and other domains. Despite these advancements, a substantial connectivity gap persists between urban and rural areas [1]. While urban areas enjoy multiple options like 5G technology and optical fiber for internet connectivity, rural areas often rely on limited choices such as 3G technology or ADSL. This discrepancy affects the availability and quality of information systems, leading to disparities in application response times and the ability to deploy quality-of-service (QoS) applications, particularly in sectors like healthcare and industry, contributing to the economic development of urban areas. Moreover, advanced technologies like fog-edge computing further widen the digital divide in urban areas. Recent efforts within the research community have aimed at developing technological architectures to extend coverage to rural areas. Community networks, exemplified by projects like [2, 3], have served as pioneering solutions. Another approach involves utilizing opportunistic mobile solutions employing store-carry-and-forward techniques [4, 5] to provide digital services in rural areas.

In recent years, researchers have explored the use of Unmanned Aerial Vehicles (UAVs) to deliver services across various domains. UAVs, with their user-friendly features and reasonable costs, have found applications in city-scale video surveillance [6], small parcel delivery [7], and disaster management monitoring [8]. The use of UAV quadcopters emerges as a promising solution for rural areas, with each UAV equipped with a small base station serving a specific area. Deploying a swarm of UAVs offers benefits like enhanced coverage, improved capacity, increased reliability, and energy efficiency [9, 10].

This work extends the research presented in [11] by introducing a UAV-based network architecture aimed at offering coverage and services to rural areas. Given the constrained computational and battery capacities of UAVs, IoT applications are broken down into microservices, with each UAV responsible for deploying and executing a particular subset of these microservices. Unlike the methodology outlined in [11], where microservice replicas are pre-deployed, the current approach dynamically assigns microservices to UAVs. This dynamic allocation minimizes energy consumption in meeting user requests by ensuring that microservices are deployed only when needed, with a predefined deployment duration. If a microservice remains inactive for a specified period, it is automatically undeployed to conserve resources.

Simulation results in a realistic scenario show that the algorithm proposed in this paper and the work published in [11] effectively reduce the energy requirements of the UAV-based network and maximize the number of IoT requests served Additionally, a comparison between the two approaches is provided. The remainder of the article is organized as follows: Sect.2 offers a review of related

works, Sect. 3 describes the system model, including the UAV-based network, microservice-based IoT application model, and power consumption model. Sections 4 and 5 present the proposed algorithms for serving IoT application requests while minimizing energy consumption. Section 6 reports and discusses experimental results, and finally, Sect. 7 provides concluding remarks based on the findings.

2 Related Work

In the pursuit of enhancing energy efficiency within Unmanned Aerial Vehicle (UAV)-based networks, various methodologies have been explored in prior research. For instance, in the study referenced as [12], a task migration strategy employing a federated Deep Q-Network (DQN) is introduced. This strategy takes into account load deviation and energy deviation among UAV Mobile Edge Computing Systems (MECs). Utilizing DQN, a local model is created for migration optimization for each UAV MECs, while federated learning generates a more robust global model by leveraging common spatial features between neighboring regions. The performance assessment of this strategy involves analyzing delay constraint satisfaction, load deviation, and energy deviation.

In another study, detailed in [13], researchers address the challenge of overcoming computation and energy resource constraints in UAVs by incorporating Mobile Edge Computing (MEC). They propose deploying container-based microservices to MEC for improved Quality of Service (QoS) by processing tasks offloaded from UAVs. The task scheduling problem is addressed, considering the deployment of new services or utilizing existing ones to balance the overhead of data transmission and microservice deployment (such as container image pulling) to minimize overall task completion time. The problem is formulated as an Integer Linear Programming (ILP) and proven to be NP-hard. An incentive-based request scheduling algorithm is also proposed.

In a different study, as presented in [14], the authors suggest the use of UAVs as an energy-efficient, reorganizable wireless network to serve Virtual Networks (VNs), in contrast to traditional cellular networks with fixed base station topology. They advocate for adaptive adjustments in the allocation of caching, computing, and communication resources, as well as the UAV-based wireless network topology, based on the service requirements of VNs, utilizing appropriate artificial intelligence algorithms. Additionally, in [15], a UAV-aided intelligent wireless sensing scheme for collecting data from IoT devices is introduced. Optimal path planning approaches for UAV deployment are proposed to minimize completion time and energy consumption, considering both peer-to-peer UAV-IoT sensing networks and clustering UAV-IoT sensing networks.

Concluding this section, authors in [16] propose a system architecture involving smart mobile devices (SMDs), unmanned aerial vehicles (UAVs), and a base station (BS). They establish a model called the Markov decision process with unknown rewards (MDPUR) to comprehensively consider migration distance, UAVs' residual energy, and load status. To achieve an effective task migration strategy, they propose

an advantage-based value iteration (ABVI) algorithm based on the MDPUR model. The algorithm aims to achieve load balancing and reduce the total energy consumption of the UAV group while maintaining user service quality.

To the best of our knowledge, this represents the first initiative aimed at proactively deploying microservices in a UAV-based network to serve rural populations requesting IoT applications with the objective of minimizing energy consumption. The subsequent section outlines the system model description for IoT application mapping in the UAV-based network architecture.

3 System Model

This section presents the system model details for allocating IoT applications within a network architecture centered around Unmanned Aerial Vehicles (UAVs). Initially, the discussion covers considerations related to the network architecture. Subsequently, the IoT application model, structured upon a microservices architecture, is delineated. Lastly, the power consumption model specific to the UAV swarm is elucidated.

3.1 UAV-Based Network Architecture

As mentioned in the Introduction, the focal setting is a remote area devoid of Internet connectivity. Consequently, a network architecture is proposed, comprising a swarm of UAVs capable of communication through wireless connectivity. Each UAV is equipped with a 5G base station, enabling communication with nearby UAVs within a specified range. This configuration ensures comprehensive coverage of the designated area by the UAV swarm, allowing users to engage in IoT applications. It is important to note that each UAV has the capacity to serve a certain number, denoted as n, of users situated within a specific radio range. Users can request the execution of specific IoT applications from the connected UAV, which must then furnish the corresponding information. The assumed Line of Sight (LoS) conditions between users and UAVs arise from the absence of significant obstacles, such as buildings or trees, that could compromise the Quality of Service (QoS).

Factors like disruptions due to adverse weather conditions or UAV failures are deferred for future exploration. To enhance the computational capabilities of UAVs and juxtapose a scenario without internet connectivity against one with cloud accessibility, it is posited that a single UAV in the network can establish a connection with a cloud node, as illustrated in Fig. 1.

3.2 UAVs Coverage and Path Loss Model

As the authors report in [9] (in the article [9]), the considered propagation pattern of the conical antenna placed on top of the UAVs has an angle of θ. A circular area is projected on the ground (see Fig. 2, (in the article Fig. 2)), whose radius, r, is proportional to the UAV altitude, h_k, according to Eq. 1:

Fig. 1 UAV-based network scenario

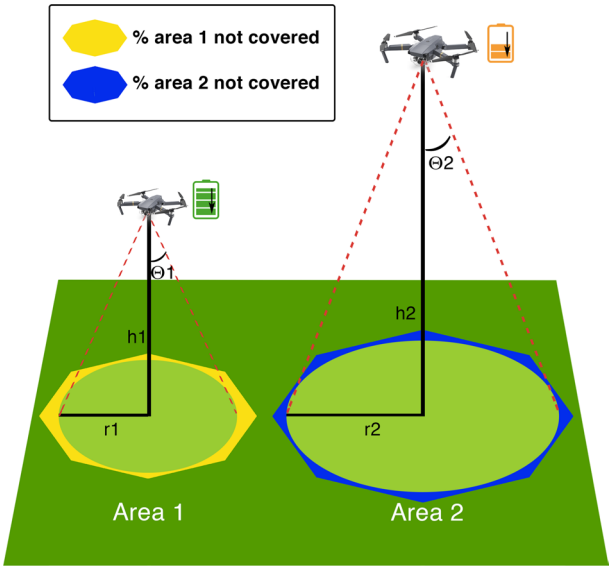


Fig. 2 Example of coverage range achieved by an UAV as a function of the altitude

$$r = h_k \cdot \tan(\theta) \tag{1}$$

where θ is the propagation pattern angle of the conical antenna, which determines the direction and amplitude of the electromagnetic radiation emitted by the antenna of the UAV and h is its current altitude. Although we assume a circular coverage by means of the UAV, the scenario is divided into a set of areas applying Voronoi Tesselation as in [9], which can result in a small overlap in terms of coverage provided by two adjacent UAVs. If we move now our focus to the path loss model that is considered, it is based on the one defined in [17], where the probabilistic mean path loss

between an UAV that is placed at altitude h_k over a location $z = (x, y)$ (where x and y are the coordinates of the point where the UAV is placed) is defined by Eq. 2:

$$\chi(h_k, r_i) = \frac{A}{1 + d \cdot \exp\left(-e \cdot \left(\frac{180}{\pi}\right) \tan^{-1}\left(\frac{h_k}{r_i}\right) - d\right)} + 10\log(h_k^2 + r_i^2) + B \tag{2}$$

where $A = \eta_{\text{LoS}} - \eta_{\text{NLoS}}$, and $B = 20 \cdot \log(\frac{4\pi f_c}{c}) + \eta_{\text{NLoS}}$. The average additional losses for LoS and No Line of Sight (NLoS) conditions are represented by η_{LoS} and η_{NLoS}, respectively. As the authors state in [18] and [19], we define both parameters on the proposed coverage model as follows: η_{LoS}, as the situation where the UAV is in η_{LoS} with the user and η_{NLoS}, when the η_{LoS} between the UAV and the user is completely obstructed by buildings. Moreover, d and e are the S-curve parameters that depend on the environment [17], f_c is the carrier frequency, r_i is the distance between the center of the UAV and the user (i), and x_i, y_i are the coordinates of the user served, as in Eq. 3:

$$r_i = \sqrt{(x_i - x)^2 + (y_i - y)^2} \tag{3}$$

We remark that the path loss model defined above is considered for the type of scenario for which the proposed solution is conceived, i.e., open rural areas in which there is a lack of interference due to the existing infrastructure and neighbouring UAVs.

3.3 Microservices-Based IoT Applications Model

In this study, we examine the scenario where users request IoT applications adhering to the Microservices Architecture (MSA). This entails decomposing IoT applications into a collection of microservices, each carrying out a specific function [20].

We define a workflow as a specific instance of an IoT application requested by a user. To elaborate, a workflow $w_u \in \mathcal{W}$ needed by user u is a well-ordered sequence of microservices $w_u = \{m_1, m_2, \ldots, m_n\}, \forall m_i \in \mathcal{M}$, which are executed consecutively to form the entire functionality of the requested IoT application. Each microservice within the workflow can potentially be deployed across various Unmanned Aerial Vehicles (UAVs), thereby involving the utilization of multiple UAVs. Consequently, the strategic placement of microservices on UAV-based computing devices becomes crucial for achieving acceptable Quality of Service (QoS), such as the maximum tolerable response time for the application. Given the NP-hard and computation-intensive nature of the service allocation problem [21], this study relies on the topological characteristics of the network topology to determine the placement of the set of microservices onto the UAVs[1].

[1] The optimal placement of microservices in the UAVs is deferred to future work.

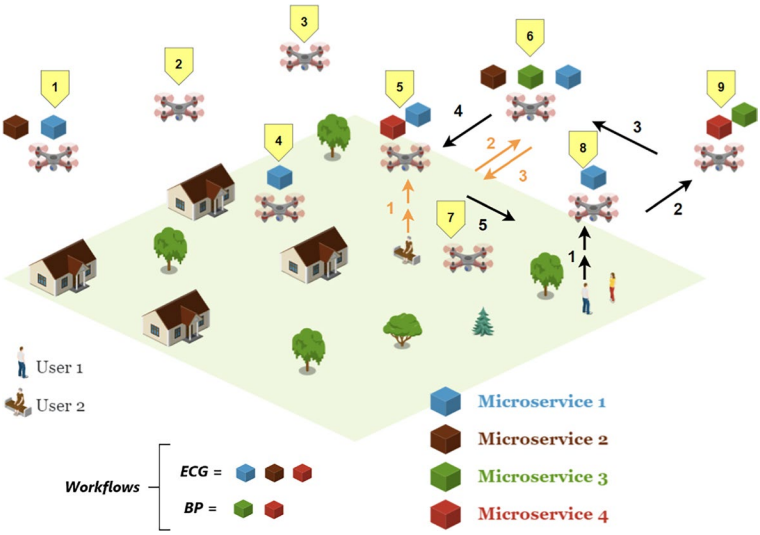


Fig. 3 Example of deployment of IoMT applications

To demonstrate the functionality of the proposed model, Fig. 3 presents a case study applicable to rural areas with limited connectivity. The scenario, illustrated in Fig. 1, is divided into nine areas, each covered by a UAV flying at an altitude of $h = 50$ m. The minimum distance between UAVs is set at 900 m to prevent interference and ensure a Line of Sight (LoS) direct link with neighboring UAVs [9]. The case study involves two users, u_1 and u_2. User u_1 is connected to a UAV and requires the execution of an electrocardiogram (ECG) IoT application, while user u_2 possesses a smartwatch and aims to monitor blood pressure, necessitating an IoT application for blood pressure (BP) analysis. Similar applications for cardiovascular disease detection are commonly employed in the field of edge intelligence [22]. Both applications adhere to the MSA, utilizing four distinct and atomic microservices: m_1 (ECG monitor), m_2 (data compression), m_3 (BP monitor), and m_4 (encryption). Consequently, the two IoT applications (ECG and BP) can be defined as workflows: $w_1 = m_1, m_2, m_4$ and $w_2 = m_3, m_4$. The ECG application (w_1) initiates with the execution of the ECG monitor microservice on the user wearable. Due to the large size of the obtained ECG data, it undergoes compression before final encryption. On the other hand, the BP application (w_2) solely requires the invocation of the BP monitor and encryption microservices. Fig. 3 depicts these two Internet of Medical Things (IoMT) applications, with associated microservices distributed on specific UAVs.

Multiple replicas of the same microservice can be deployed on different UAVs, enabling users to execute them from various parts of the scenario. Deploying a single instance of each microservice in the network would limit the computing capabilities of UAVs, leading to a reduction in the number of accepted user requests. In case of significant computing power, this would result in a reduction of latency, thus improving overall performance. Deploying replicas in different parts of the network minimizes latency, as each user can request the nearest microservice. In Fig. 3, user

u_1 requests ECG analysis, following the path represented by black arrows (UAV 8, UAV 6, and UAV 5), while the path for BP analysis requested by user u_2 is shown with yellow arrows (UAV 5 and UAV 6). It's worth noting that microservice 4 is replicated at multiple nodes and is requested by both applications.

3.4 Power Consumption Model

In this section, the description of the energy consumption model is provided. The energy consumption to process a microservice m_i^n by UAV n aggregates the energy consumption required by the execution of the microserve itself plus the energy consumption derived from the transmission of the outputted data to the next UAV in the chain. Thus, if we denote with $\mathcal{E}_{i,n}^{\text{PROC}} = \sum_{k=1}^{c_i} p_n \cdot f_n$ as the energy required by UAV n to process and execute microservice i, and the energy required by UAV n to send the information to the next UAV in the sequence (if any) or to the end user if it is the last one in the workflow as $\mathcal{E}_{i,n}^{\text{TRANS}}$, then the energy consumption required to execute a workflow w_u requested by user u that requires the execution of K microservices in a subset of UAVs $\mathcal{N}' \subseteq \mathcal{N}$ can be assessed applying Eq. 4.

$$\mathcal{E}_{w_u} = \sum_{i=1}^{K} \mathcal{E}_{i,n}^{\text{PROC}} + \sum_{i=1}^{K} \mathcal{E}_{i,n}^{\text{TRANS}} \qquad \forall n \in \mathcal{N}' \tag{4}$$

The definition of energy required for transmission for one UAV is defined according to Eq. 5:

$$\mathcal{E}_{i,n}^{\text{TRANS}} = E_{i,n} \times \left(\frac{1000}{V} \right) \tag{5}$$

The remaining components of the equation are explained as follows according to Eqs. 6 and 7:

$$t_i = \frac{L_i}{R} + \frac{d}{s} \tag{6}$$

$$E_{i,n} = p_n \times t_i \tag{7}$$

where E_b represents the UAV battery consumption, V represents the Battery nominal voltage, t_i represents the time taken to transmit the information between one node to the next node in the path used in the communication process, L_i represents the input size of the microservice to be transmitted, this means, sending or transferring the information related to the microservice from one UAV to another, R represents the transmission speed in the channel, d represents the minimum distance among UAVs and s represents the speed of light in vacuum and p_n represents the power coefficient of UAV n. Table 1 shows the notation used in the system model. In this way, each of the equations involved in the transfer cost of a specific microservice from one UAV to another in the network would be defined.

Table 1 Symbols notation

Symbol	Description
$\mathcal{N}$	Set of UAVs
$\mathcal{L}$	Set of wireless links connecting $\mathcal{N}$
$\mathcal{U}_n$	Set of users connected to UAV n
z_n	Location (x_n, y_n) of UAV n
$\mathcal{M}_n$	Set of microservices deployed at UAV n
f_n	CPU frequency of UAV n
p_n	Power coefficient of UAV n
w_u	Workflow of IoT application requested by user u
m_i^u	Microservice in the i-th position of workflow w_u
$\mathcal{E}_n^{\text{RES}}$	Remaining battery of UAV n
$\mathcal{E}_{i,n}^{\text{PROC}}$	Power consumption to process microservice i at UAV n
$\mathcal{E}_{i,n}^{\text{TRANS}}$	Power consumption to transfer microservice i from UAV n to the next hop
$\mathcal{E}_{w_u}$	Power consumption required to execute workflow w_u requested by user u
L	Input size of the microservice to be transmitted
R	Transmission speed in the channel, in this case using WiFi
d	Minimum distance among UAVs
s	Speed of light in vacuum
V	Battery nominal voltage [23]
t	Transmission time
E_b	UAV battery consumption
c_i	Represents the CPU cycles required by the microservice i

Note that there may happen that two consecutive microservices in the workflow can be deployed in the same UAV. In this case, $\mathcal{E}_{i,n}^{\text{TRANS}} = 0$, since there is no need to send the data toward a different UAV.

3.5 Objective Function

After having defined all the models considered in our work, this section describes the objective function. Let us consider a network graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ composed of a set of UAVs $n \in \mathcal{N}$ hovering at altitude h over a location $z_n = (x_n, y_n)$. Each UAV n is connected to a subset of neighbours $\mathcal{N}^n \subset \mathcal{N}$ according to WiFi technology. A set of users $\mathcal{U}_n$ is connected to each UAV $n \in \mathcal{N}$ and require to execute requests for different IoT applications. Moreover, each UAV has the capability of providing computing resources, thus deploying and invoking a set of microservices $\mathcal{M}_n \subset \mathcal{M}$.

Each microservice $m_i \in \mathcal{M}$ is defined as $m_i = (c_i, d_i)$, where c_i represents the CPU cycles required by the microservice and d_i is the data associated it. Thus, an UAV n is defined by a 5-tuple of type $n = \{z_n, \mathcal{M}_n, f_n, p_n, \mathcal{E}_n^{\text{RES}}\}$, where z_n is the position of the UAV, M_n indicates the set of deployed microservices, f_n represents the CPU frequency, p_n is the power coefficient and $\mathcal{E}_n^{\text{RES}}$ refers to the remaining battery of the UAV. The objective function, which aims at minimizing the energy consumption

required to serve all the IoT requests composed by workflows, and required by all the users in the scenario, is defined according to Eq. 8:

$$min \sum_{u}^{\mathcal{U}} \mathcal{E}_{w_u} \tag{8}$$

After the definition of all the models considered in the architecture and the problem description, next section describes the algorithm that is proposed to minimize the energy consumption required by the execution of IoT applications in rural scenarios where coverage is provided exploiting an UAV-based network architecture.

4 EEMSIoT Algorithm

In this section, we provide an overview of the Energy-Efficient algorithm for MicroService-based IoT applications in rural areas (EEMSIoT). The primary objective of EEMSIoT is to minimize the energy consumption of Unmanned Aerial Vehicles (UAVs) while catering to IoT applications in a scenario where microservices are distributed across a UAV-based network. As previously mentioned, for each IoT request made by a user, the algorithm seeks to determine a path from the source UAV (the one to which the user is connected) to the set of microservices that need to be executed by the requesting application. The application specifies the order of microservices execution, and the network may have varying numbers of replicas for each type of microservice.

Algorithm 1 EEMSIoT pseudo-code

Require: The network topology $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, the source node: s, the workflow associated to the application request from user u: w_u

1: $path \leftarrow \emptyset$
2: $min\_cons \leftarrow \infty$
3: $n \leftarrow s$ ▷ Current UAV
4: **for all** $m \in w$ **do** ▷ Handle each microservice
5: $best\_path \leftarrow \emptyset$
6: $paths \leftarrow$ shortestPath(n, m) ▷ Find the shortest path from node n to all nodes hosting microservice m
7: **for all** $p \in paths$ **do**
8: $isOk \leftarrow$ checkConstraints(p, k, m) ▷ k is the node where microservice m is deployed
9: **if** $isOk$ **then**
10: $cons \leftarrow$ assessEnergyConsumption(n, k, p)
11: **if** $cons \leq min\_cons$ **then**
12: $min\_cons \leftarrow cons$ ▷ Update of minimum energy consumption
13: $best\_path \leftarrow p$ ▷ Update of the best path
14: **end if**
15: **end if**
16: **end for**
17: $n \leftarrow k$ ▷ Update of current UAV
18: $path \leftarrow path \cup best\_path$
19: **end for**
20: $return_path \leftarrow$shortestPath(n, s) ▷ Find the shortest path back to the source node
21: **return** *path, return_path*

Given that the primary goal is to manage user application requests and minimize overall energy consumption, EEMSIoT relies on the shortest path technique to determine the placement of the next microservice in the application workflow. When faced with multiple options for selecting the next microservice, the algorithm prioritizes the one deployed in the Unmanned Aerial Vehicle (UAV) with the highest remaining battery. It is essential to adhere to certain constraints: (i) the UAV must possess sufficient battery capacity to fulfill the request, and (ii) the UAV must have enough processing capability to execute the microservice[2]. Once a suitable UAV is chosen to execute the current microservice, the algorithm proceeds to the next microservice in the workflow. After completing the workflow, the algorithm must find a return path back to the source UAV to fulfill the request. Should a request be unable to be fulfilled due to battery or computation constraints, the algorithm explores the possibility of leveraging cloud capabilities. In such cases, the algorithm aims to reach the UAV connected to the cloud (with internet access), as illustrated in Fig. 1.

Algorithm 1 outlines the pseudo-code of EEMSIoT. It begins with an initialization phase and then, starting from node n, aims to identify the set of shortest paths from n to the UAVs containing a replica of microservice m (line 6). For

[2] If the next microservice in the workflow is also deployed in the current UAV and satisfies both battery and processing constraints, it can be executed within the same UAV.

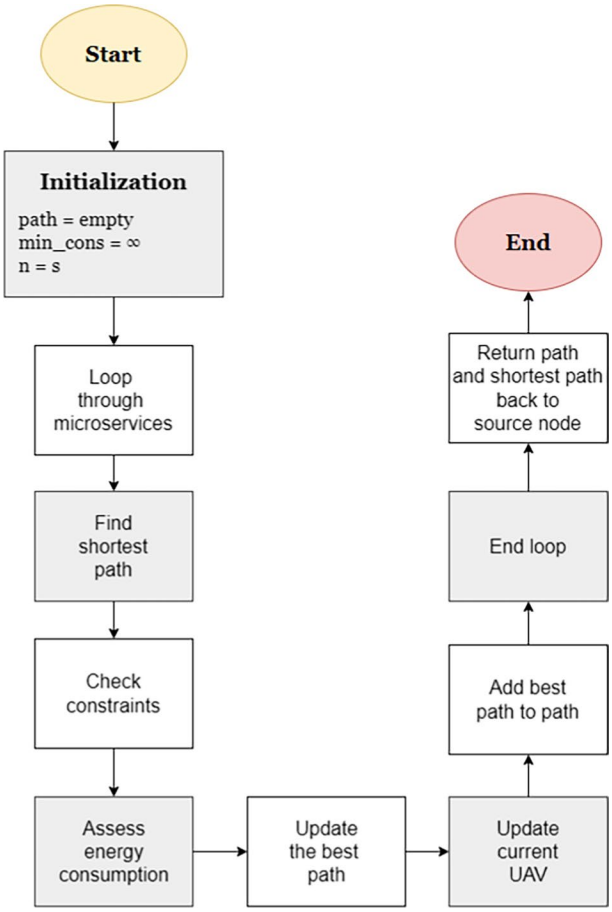


Fig. 4 Flow diagram for EEMSIoT algorithm

each potential path, the algorithm verifies the remaining battery and processing capacity of each UAV (line 8), it allows to check if it is possible to build a path that meets the battery and memory constraints, which allows to route the information of the requested microservice (m) deployed at the point k to the node n. If these constraints are met, is calculated the cost of sending the information from node k to node n, the initial node where it is requested, through the previously calculated path (line 10), and then, the algorithm selects the path with the minimum energy consumption - specifically, the one with the UAVs having the highest remaining battery from node n to the selected UAV k hosting m (lines $9-16$). This process is repeated for each microservice in the workflow. Finally, a return path is evaluated to reach the source node where the request was initiated.

To conclude the explanation of the EEMSIoT algorithm, Fig. 4 presents the flow chart depicting the main functionalities and decisions involved in its implementation.

5 UAV-GreenMS Algorithm

In this section, we tackle the microservices placement problem at UAVs by proposing a new solution to minimize the overall network energy consumption. Differently from EEMSIoT, for which it is assumed that the set of microservices are already deployed in the set of UAVs, a novel algorithm, namely UAV-GreenMS, is proposed. This algorithm is defined to achieve an higher scalability and a lower energy consumption in UAV-based networks by trying to (near) optimally deploy microservices according to user requests.

Keeping in mind that the objective is to handle the application requests from users and to reduce the overall energy consumption, UAV-GreenMS is based on the shortest path technique to find the best placement of the next microservice in the application workflow. If there are several options to select the next microservice, the algorithm will calculate the cost in two possibles situations: (i) the cost to deploy the microservice and its dependencies (if it has) in the UAV; and (ii) the cost to transmit the microservice requested from the UAV Network, i.e., another UAV where the microservice is already deployed. Once both costs have been calculated, the algorithm will select the situation in which the cost is lower. Clearly, several constraints must be satisfied for this decision: (i) the UAV has the battery level above a predefined threshold and sufficient battery to either transmit a microservice both to the user and within the network, or deploy the requested microservice on itself; and (ii) the UAV has enough processing capacity to run the microservice[3].

[3] If the next microservice in the workflow is also deployed in the current UAV and it does not violate both battery and processing constraints, the microservice can be executed within the same UAV.

Algorithm 2 UAV-GreenMS pseudo-code

Require: The network topology $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, the source node: s, the workflow associated to the application request from user u: w_u

$n \leftarrow s$ ▷ Current UAV
$s\_path \leftarrow \emptyset$
for all $m \in w$ **do** ▷ Handle each microservice
  if $m \notin listMicroservices(n) \vee listMicroservices(n) = \emptyset$ **then**
    $costInUAV \leftarrow$ calculateCostInUAV(n, m)
    $costFromNetwork \leftarrow$ costFromNetwork$(\mathcal{G}, m, s_path)$
    if $costFromNetwork \neq 0$ **then**
      $deploy \leftarrow$ checkDeployConstrains(n, m)
      if $costInUAV \leq costFromNetwork \wedge deploy$ **then**
        deployMicroservice(n, m)
      else
        $trasmit \leftarrow$ transmitConstrains$(\mathcal{G}, m, s_path)$
        if $transmit$ **then**
          executeTransmision$(\mathcal{G}, m, s_path)$
          updateUAVsBatteriesInPath(m, s_path)
          updateMicroservicesTimeInPath(m, s_path)
        else
          showDenegationMessage() ▷ UAV cant give the request
        end if
      end if
    else
      $deploy \leftarrow$ checkDeployConstrains(n, m)
      if $deploy$ **then**
        deployMicroservice(n, m)
      else
        showDenegationMessage()
      end if
    end if
  else
    $enoughBattery \leftarrow$ checkBatteryConstrains(n, m)
    if $enoughBattery$ **then**
      updateUAVbattery(n, m)
      updateMicroservicesTimeDeployed(n, m)
    else
      showDenegationMessage()
    end if
  end if
  updateMicroservicesTime$(\mathcal{G}, n)$ ▷ Update microservices time deployed in the network
end for

In UAV-GreenMS, a maximum deployment timeout is set for each already deployed microservice on an UAV. Thus, a deployed microservice has not received a request for some timeout, it can be removed from the set of microservices availables in the UAV: (i) if the microservice reaches that timeout, it must be removed from the set of microservices deployed on the UAV; and (ii) if the microservice is requested in the UAV before this timeout is expired, it is reset. For both deployment and transmission of a microservice, it is also necessary to satisfy the constraints of the UAV in question mentioned above (battery

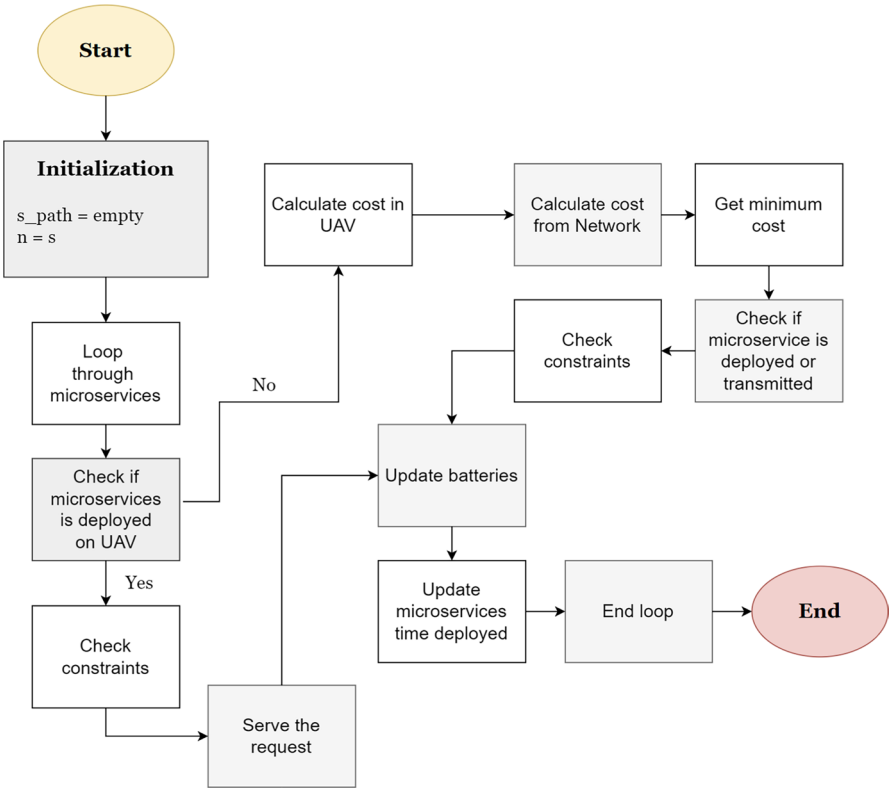


Fig. 5 Flow diagram for UAV-GreenMS algorithm

and computation). Once the available UAV is selected to transmit or deploy the requested microservice, the algorithm will move on to the next microservice in the workflow. If a request cannot be satisfied due to battery or computation constraints, the request will be rejected.

Alg. 2 shows the UAV-GreenMS pseudocode. First, an initialization phase is performed. Then it is checked if the microservice is not deployed on the current node n (line 4) or if it is already deployed (line 29).

In the first case, where the microservice is not deployed on the current node, the cost of deploying the microservice on the current node and the cost of transmitting it from another UAV on the network which has deployed the microservice, are calculated. Then, we check if the requested microservice m is available on any of the nodes in the network. If it is available (line 7), we compare the cost of deploying it on the current node or transmitting it from the network (line 9). In case the microservice is not available, the battery and resource constraints required for its deployment are checked (lines 22–23). If the constraints are satisfied, the microservice m will be deployed on node n (line 24), otherwise the microservice request will be rejected. As mentioned earlier, if the cost of

Fig. 6 UAV network deployment over the considered scenario

deploying microservice *m* on the current node is lower than the cost of transmitting it from the network and the node *n* meets the deployment constraints (line 9), microservice *m* will be deployed (line 10). On the other hand, if the cost of requesting and transmitting the microservice from any of the UAVs in the network is lower or if the node *n* does not meet the deployment constraints, it is checked whether the node *n* can satisfy the request by requesting this microservice from the network (lines 12–13). If the constraints are met, the transmission would be executed along the calculated shortest path, from the UAV where the requested microservice is deployed, to the UAV where the microservice has been requested. *s_path*, and the battery of the UAVs involved in the transmission would be updated as well as the timeout of the requested microservice *m* on each of the UAVs involved in the *path* (lines 14–16). Conversely, if the node *n* does not meet the constraints for transmitting the request, it would be rejected (line 18).

In the second case, where the requested microservice *m* is already deployed on the current node *n* (line 29), it is checked whether the current node has enough battery to continue maintaining such deployed microservice (line 30). If it does, the nodes battery and the microservices deployed timeout are updated (lines 32–33). On the other hand, if the node cannot maintain the deployed microservice, the request will be rejected. At the end of the execution, the timeouts of all deployed microservices in the network are updated (line 38).

Table 2 Microservices specification

Microservice	Description	RAM	Battery	Input Size
m_1	ECG monitoring	393 MB	24.4 GCycles	3.93 MB
m_2	Compression	136 MB	9.9 GCycles	1.36 MB
m_3	Blood pressure monitoring	393 MB	24.4 GCycles	3.93 MB
m_4	Encryption	79 MB	6.1 GCycles	0.79 MB

To conclude the explanation of the UAV-GreenMS algorithm, the flow chart illustrating the main functionalities and decisions addressed in its implementation is reported in Fig. 5.

5.1 Complexity Analysis

In the following, a complexity analysis of EEMSIoT and UAV-GreenMS algorithms is provided. Two variables must be taken into account for the analysis: the number of UAVs that are part of the network, $\mathcal{N}$, and the number of requests carried out by users, U, i.e. the number of users interacting with the system. In the case of the UAV-GreenMS algorithm (Alg. 2), the functions to update the deployment time of the microservices require $\mathcal{O}(U \cdot \mathcal{N}^3)$. On the other hand, the assessment of the battery consumption in the EEMSIoT algorithm (Alg. 1) requires $\mathcal{O}(U \cdot \mathcal{N}^3)$. Finally, in both algorithms, functions such as the calculation of the shortest path or the deployment of microservices require $\mathcal{O}(U \cdot \mathcal{N}^2)$. This means that the resulting complexity for both proposed algorithms is $\mathcal{O}(U \cdot \mathcal{N}^3)$.

6 Experimental Results

In this section, EEMSIoT and UAV-GreenMS algorithms are evaluated and compared through simulations over a realistic rural scenario. First, the simulation set-up is described, including the considered scenario and the parameter setting. Then, an evaluation of the number of satisfied IoT requests as a function on the users density is performed. Next, an analysis on the average remaining battery per UAV is carried out, considering the variation in the users density and the percentage of microservices that are deployed. Finally, the impact of users distribution is also evaluated.

6.1 Simulation Set-Up

EEMSIoT and UAV-GreenMS have been evaluated over a realistic rural scenario located in Valle del Jerte, Cáceres (Spain), which is composed of 5 villages (Casas del Castañar, Cabrero, Barrado, Piornal and Valdastillas) connected by rural roads as it can be seen in Fig. 6. In order to provide coverage and services to the population of such villages, an UAV-based network architecture based on the description of Sect. 3 is deployed over the scenario by placing a set of UAVs at an altitude

of 50 m. over the ground, with a maximum distance among adjacent UAVs of 900 m [9]. With the selection of this setting, interference in the inter-UAV communication is minimized. Note that both algorithms are executed on an external server that will perform the function of network controller, thus allowing each of the functions developed in each of the algorithms to be carried out. Moreover, LoS conditions are assumed between users and UAVs, since the considered scenario lacks of big obstacles (buildings, trees, etc.) that could degrade the QoS obtained. Disruptions due to bad weather conditions or UAV failures are left for future work. Thus, the considered network is composed of 57 UAVs that provide coverage to the residential areas and to the roads connecting them. Fig. 6 shows the UAVs deployment over the considered rural scenario.

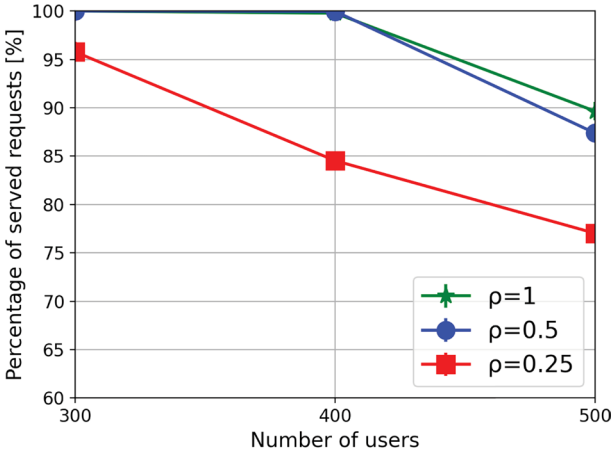
Each UAV in the given scenario is furnished with a Raspberry Pi boasting 4 GB of RAM, a 1 GHz CPU, and a 1000 W/h battery. These specifications, sourced from [24, 25], serve as the computing and processing capacities to deploy the necessary microservices. To assess the outcomes of the proposed algorithm in an UAV-based network lacking Internet connectivity, a comparison is made with the option of connecting the UAV swarm to the cloud. A ground base station, functioning as a cloud node with 16 GB of RAM, is introduced for augmenting computing capabilities. Consequently, one of the UAVs positioned at the network's edge can establish a connection with the cloud node. This UAV is tasked with forwarding Internet of Things (IoT) requests to the cloud node if the computing load of the UAV-based network nears maximum capacity, and no other UAVs are available for processing.

To oversee the health parameters of elderly individuals residing in the given setting, two applications within the realm of the Internet of Medical Things (IoMT) have been examined. The first is an Electrocardiogram (ECG) IoMT application that involves the implementation of a workflow consisting of three microservices, denoted as $w_1 = \{m_1, m_2, m_4\}$. The second is an IoMT application dedicated to monitoring blood pressure (BP), requiring the execution of two microservices, namely $w_2 = \{m_3, m_4\}$. Additionally, other pertinent monitoring applications, such as those focused on cardiovascular disease detection, are employed within the context of edge intelligence [22].

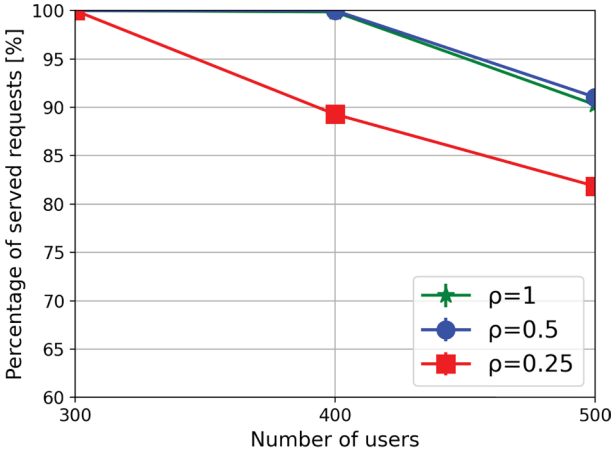
Table 2 shows the RAM requirements and the battery cycles per microservice, according to the values reported in [24–26]. Concerning the power consumption derived by the use of microservices, each cycle requires an amount of $3.5 \cdot 10^{-9}$ W, while $2 \cdot 10^{-4}$ W are required per each second employed in a request [27].

To assess the impact of the number of replicas per microservice in the given scenario, particularly for the EEMSIoT algorithm, we introduce a parameter denoted as ρ, representing the percentage of replicas for each type of microservice deployed in the UAV-based network. A value of $\rho = 1$ signifies one replica per microservice at each UAV for all microservice types. We explore three distinct scenarios with ρ taking values $\{0.25, 0.5, 1\}$ to examine various deployment situations.

When $\rho < 1$, the method employed to select the subset of UAVs for deploying microservice replicas becomes crucial and can significantly impact the algorithm outcomes, especially in terms of Quality of Service (QoS) for IoMT applications. Our approach relies on leveraging topological network features to guide the microservices placement method. Through experimentation, we have confirmed



(a) Lack of cloud



(b) With cloud

Fig. 7 Percentage of served requests for EEMSIoT as a function of the number of users in the scenario

that the *Highest Node Degree* method, where replicas are deployed on UAVs with the highest number of neighbors (i.e., wireless links to other UAVs), yields the most favorable outcomes. In this way, a value of $\rho = 0.5$ means that there will be one replica of each microservice per UAV in 50% of the total number of UAVs in the network, and a value of $\rho = 0.25$ means that there will be one replica of each microservice per UAV in 25% of the total number of UAVs in the network. This selection is motivated by the fact that such nodes are potentially sought after by other nodes to handle offloaded requests and invoke microservices belonging to the requested application workflow. It is worth noting that the UAV-GreenMS

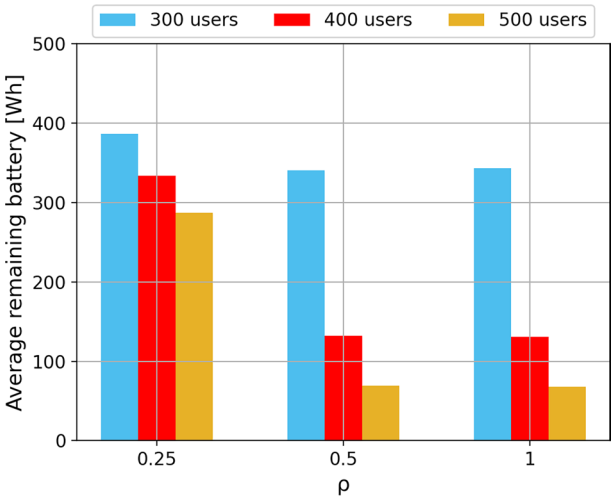


Fig. 8 UAVs average remaining battery for EEMSIoT

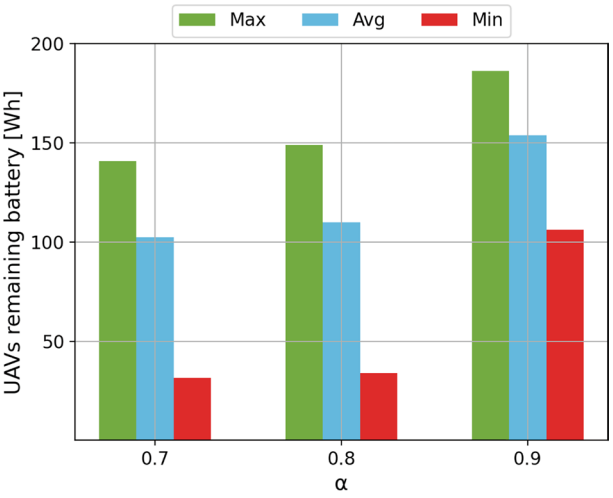
algorithm does not require these parameters, as microservices are not initially deployed in the UAVs; instead, they are deployed or requested at runtime.

For the evaluation of the scalability of the solution within the defined scenario, we conduct simulations with varying numbers of users who may potentially request IoT applications. Additionally, we explore different user placement distributions. Given that a significant portion of the population resides in villages, the predominant users are assumed to be connected to UAVs stationed in these villages. The remaining percentage of inhabitants is presumed to travel via roads, connecting to UAVs along the paths between villages. This simulation reflects a realistic scenario where the majority of individuals (parameterized as α) are situated in rural locations, while a smaller percentage of users (parameterized as β) are traveling by car. Three scenarios are considered: ($\alpha = 0.7, \beta = 0.3$), ($\alpha = 0.8, \beta = 0.2$), and ($\alpha = 0.9, \beta = 0.1$). Notably, the selection of the type of IoT application per user is carried out randomly.

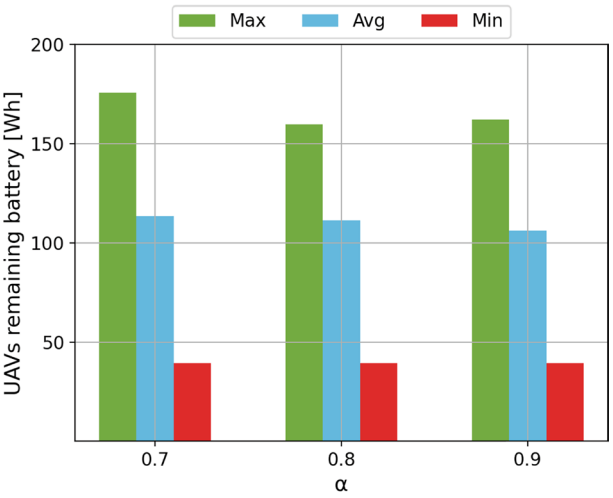
6.2 Performance Evaluation of EMMSIoT

In this section, we conduct an evaluation of EEMSIoT , with a focus on two primary metrics: (i) the percentage of served IoT application requests and (ii) the remaining battery of UAVs after fulfilling the set of requests for users.

Fig. 7 illustrates the percentage of IoT application requests that EEMSIoT can handle as a function of the number of users in the scenario, considering different percentages of replicas of microservices (ρ). We evaluate two situations: (i) a complete lack of connectivity (Fig. 7a), and (ii) a scenario where a single UAV can reach the cloud, thereby extending computing capabilities (Fig. 7b). There is a clear decreasing trend in the number of served requests as the number of users requesting IoT applications increases. This is attributed to the fact that with more users seeking networking and computing capabilities, there are fewer opportunities to address



(a) $\rho = 1$



(b) $\rho = 0.5$

Fig. 9 UAVs remaining battery for EEMSIoT as a function of the users distribution for 400 users

their needs. Additionally, if there is an option to have a cloud node connected to a UAV, tasks can be offloaded to the cloud, thereby increasing the possibilities of serving more IoT requests.

Note that in the scenario with a cloud node to which offload the requests (Fig. 7b), the distribution of the microservices for the case of $\rho = 0.5$ following the *Highest Node Degree* approach could be more concentrated in the areas where users are located, so that it could present slightly better results than the case of $\rho = 1$, for which it could happen that some of the UAVs could run out of battery or memory.

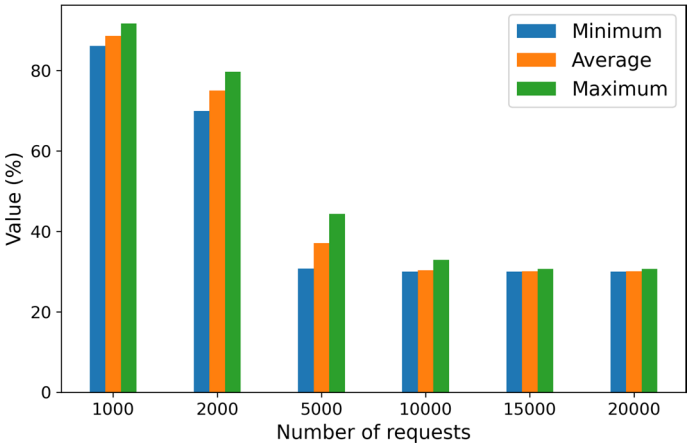


Fig. 10 UAVs battery level percentage for UAV-GreenMS

Regarding the proportion of replicas for each type of microservice deployed across the network, it is evident that the algorithm encounters difficulties when this number is low (indicated by the red line). However, as the number of replicas increases, the results improve, stabilizing around 100% when $\rho = 0.5$ and $\rho = 1$ with 400 users. Another noteworthy point is that, despite a reduction in the percentage of served requests when the user count rises to 500 (attributed to an influx of IoT requests exceeding UAV handling capacity), there are no significant variations observed when comparing $\rho = 0.5$ and $\rho = 1$. This underscores the notion that deploying all microservices on every UAV is not necessary; instead, an appropriate selection of a subset ($\rho = 0.5$) proves sufficient to maintain an acceptable Quality of Service (QoS).

If we shift our focus to the energy needed by the system to handle the previously analyzed requests, Fig. 8 illustrates the average remaining battery per UAV after addressing all the IoT requests made by the user set, considering different values of ρ. It is evident that as the number of users requesting IoT applications increases, the energy required to fulfill these requests also rises, resulting in a decreased remaining battery for UAVs. The variation in the remaining battery of UAVs becomes noticeable when the number of replicas of microservices across the network is increased. More replicas offer more execution possibilities for microservices, reducing the number of UAVs needed to reach the target microservice. However, if all UAVs have the capability to execute a microservice and they are all executed ($\rho = 1$), the average remaining battery decreases. In the case of $\rho = 0.25$, where only 25% of UAVs contain replicas, the primary energy consumption for these UAVs comes from transmitting information to the next UAV rather than the computation of microservices themselves. Nonetheless, minimal differences are observed when comparing the outcomes of $\rho = 0.5$ and $\rho = 1$, suggesting that not all UAVs need to replicate microservices but rather a subset of them.

In the following, an examination of user distribution and its impact on the remaining battery of UAVs. Our objective is to assess how the percentage of users located

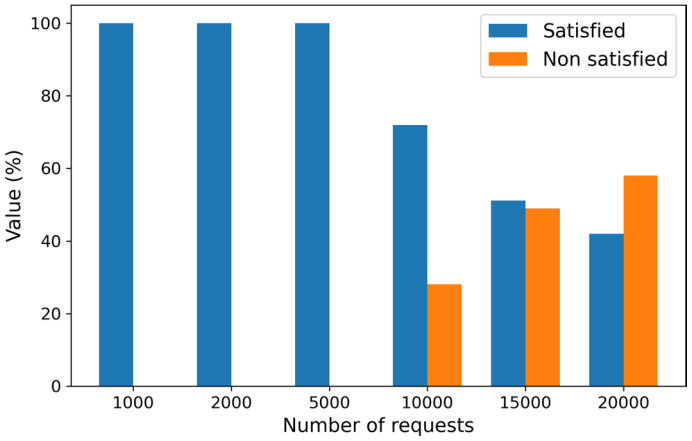


Fig. 11 Satisfied and non satisfied requests in each of the scenarios

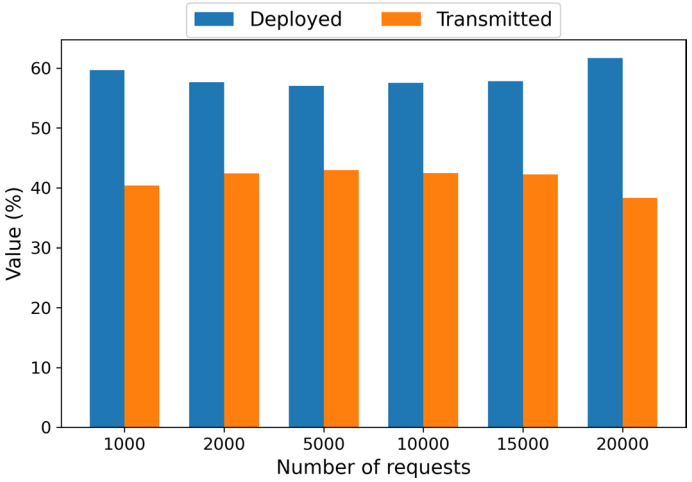


Fig. 12 Transmitted and deployed requests in each of the scenarios

in villages versus those traveling along roads affects UAV battery levels. Specifically, we focus on a scenario where 400 users request IoT applications, representing the point at which 100% of requests are served, as indicated in the analysis linked to Fig. 7. More specifically, Fig. 9a illustrates the minimum, average, and maximum remaining battery at UAVs for three user distribution cases: ($\alpha = 0.7, \beta = 0.3$), ($\alpha = 0.8, \beta = 0.2$), and ($\alpha = 0.9, \beta = 0.1$), along with the scenario of 100% replicas per microservice ($\rho = 1$). The figure highlights that when a majority of users are concentrated in villages ($\alpha = 0.9$), the remaining UAV battery is higher compared to a situation where a significant percentage of users travel through roads to reach other villages ($\alpha = 0.7$). This suggests that IoT requests are generally completed within a shorter range inside the village, leading to reduced energy consumption for serving

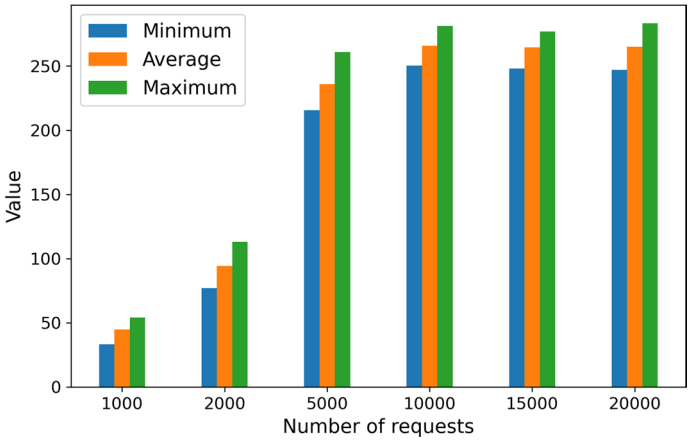


Fig. 13 Number of hops for transmission in each of the scenarios

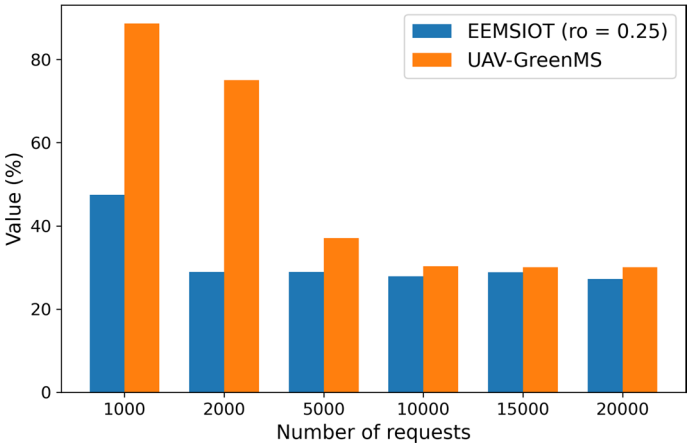


Fig. 14 Average remaining battery for EEMSIoT and UAV-GreenMS

IoT requests. Conversely, if a workflow requires reaching a UAV located on a road, the energy required is higher. A different scenario emerges when the percentage of replicas is $\rho = 0.5$. Fig. 9b presents similar results for $\alpha = 0.9$ and $\alpha = 0.7$, i.e., there is a negligible impact comparing the case in which users are mostly located in villages or traveling along roads. This is expected because the method used to place replicas at UAVs is based on their degree (number of neighbors). UAVs in villages are selected first, while those over roads are rarely reached unless the computing or battery capabilities of UAVs in villages are exhausted. This finding complements previous analyses where both the percentage of served requests and the average remaining battery were similar for $\rho = 0.5$ and $\rho = 1$. In this case, user distribution significantly influences the remaining UAV battery, particularly due to the microservices placement selection method. In the future, we plan to optimize the placement

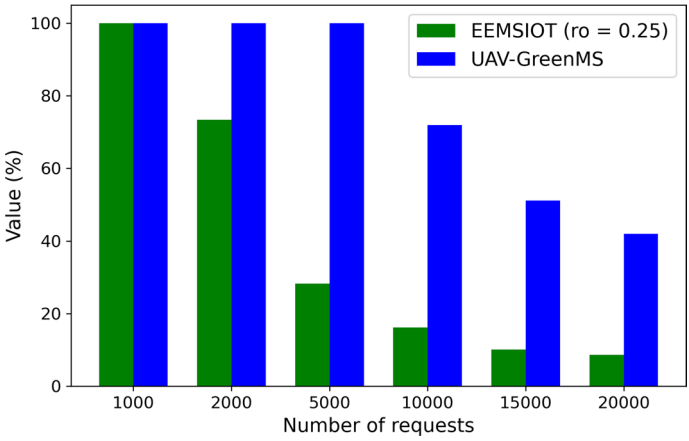


Fig. 15 Percentage of satisfied requests for EEMSIoT and UAV-GreenMS

of microservices to minimize energy requirements while maximizing the number of served IoT requests.

6.3 Performance Evaluation of UAV-GreenMS

In this section, an evaluation of UAV-GreenMS algorithm is carried out focusing on four main metrics: (i) the percentage of remaining battery after serving the set of requests from users, (ii) the number of times an UAV is involved in the transmission of a microservice to/from another UAV, (iii) the percentage of satisfied and unsatisfied requests in each scenario; and (iv) the percentage of transmitted and deployed requests in each scenario. In this case, the area under evaluation is a 3×3 submatrix covering the village Casas del Castañar, i.e., nodes 1 to 9 in Fig. 6. The number of requests from users ranges from 1000 to 20,000 and, in each simulation, requests are randomly distributed over the set of UAVs in the scenario.

First, Fig. 10 reports the minimum, maximum and average percentage of remaining battery in the set of UAVs as a function of the number of requests in the scenario. Clearly, as the number of requests is increased, the average remaining battery at UAVs starts to decrease, with a stabilization after handling 5000 requests. If we move now our attention to the percentage of satisfied requests, Fig. 11 illustrates that beyond 5000 requests, the network experiences saturation accompanied by a minor proportion of rejected requests. It is also possible to observe that, for a larger number of requests, the swarm of UAVs is not able to satisfy all requests. As the scenarios are randomly generated, the attainment of microservices can affect battery consumption, as it impacts the time in which microservices are deployed on a given UAV or not.

Fig. 12 shows the results extracted after evaluating the percentage of requests transmitted and deployed in each of the scenarios. It can be seen that both the percentage of requests directly deployed in UAVs (blue bars) and those transmitted

from UAVs to other UAVs (orange bars), remain stable with a variation of 5%, despite the increase in the number of requests.

One final point to consider is the evaluation of the number of hops in which the UAVs of the network have been involved. Fig. 13 represents the minimum, maximum and average number of hops in which the UAVs of the network have been involved in each of the scenarios. It can be seen that the number of hops in which the UAVs of the network have been involved starts to increase gradually until reaching a value of 10,000 requests. Once this is value is reached, the number of hops remains constant.

6.4 EEMSIoT and UAV-GreenMS Comparison

Having analysed the behaviour of each of the two proposed algorithms, several things can be stated: (i) performing a microservice placement on UAVs is more effective in terms of energy consumption compared to the case in which all the microservices are already deployed, even considering the topological features of the network for the deployment.

In Fig. 14, it can be seen that UAV-GreenMS outperforms EEMSIoT for $\rho = 0.25$ regardless the number of user requests. For a small number of requests such as 1000 requests, the average remaining battery of UAVs in UAV-GreenMS is nearly the double compared to EEMSIoT. If this number is increased, the gap among the two solutions is stabilized, being always higher for UAV-GreenMS; ii) On the other hand, placing microservices in the UAVs allows obtaining a higher percentage of satisfied requests than having the microservices already deployed in the UAVs. In Fig. 15, it can be seen that UAV-GreenMS again presents the best outcomes compared to EEMSIoT, regardless of the number of user requests. For a small number of requests, such as 1000, both algorithms are capable of satisfying 100% of the requests from users. However, once this threshold is exceeded, the percentage of satisfied requests decreases drastically for EEMSIoT and slightly for UAV-GreenMS, always being favorable for UAV-GreenMS algorithm.

7 Conclusion

Rural zones are less desirable and attractive for network operators, since the low density population does not justify the deployment of the required infrastructure to provide broadband Internet access. As a result, the lack of connectivity prevents the deployment of key services such as remote health-care, emergency services, or remote learning. Fortunately, UAV-based networks represent a promising solution to reduce the coverage gap in the world. In order to bring services closer to people living in rural areas, this paper exploits the capabilities of UAV-based networks to propose an energy-efficient solution to deploy microservice based IoT applications that can improve the quality of life of rural population. Simulation results show the effectiveness of the proposed solution, evaluating the percentage of IoT requests that are served to users in a realistic scenario and reducing the energy consumption

required by UAVs when handling such requests. As future research steps, we plan to evaluate the proposed solution over a real test-bed and in a real rural scenario.

Author Contributions All authors contributed to the study conception and design. Material preparation, simulation and analysis were performed by BP. All authors read and approved the final manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work has been partially funded by MCIN/AEI/10.13039/501100011033 and by the European Union "Next GenerationEU /PRTR", by the Ministry of Science, Innovation and Universities (projects TED2021-130913B-I00, PDC2022-133465-I00), by the by the Cap4IE project (0786_CAP4ie_4_P) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the project PID2021-124054OB-C31 and the grant CAS21/00057 (MCI/AEI/FEDER, UE), and by the Regional Ministry of Economy, Science and Digital Agenda of the Regional Government of Extremadura (GR21133).

Data Availability This manuscript has no associated data file.

Declarations

Conflict of interest The authors declare no competing interests, financial or otherwise.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Delaporte, A., Bahia, K.: The state of mobile internet connectivity 2021. Technical report, GSMA Connected Society (2021)
2. Fon is the global WiFi network with millions of hotspots. https://fon.com/. Accessed 31 May 2023
3. Talbot, D., Hessekiel, K., Kehl, D.: Community-owned fiber networks: Value leaders in America. Technical report, Berkman Klein Center for Internet & Society Research Publication (2017)
4. Jesús-Azabal, M., Herrera, J.L., Laso, S., Galán-Jiménez, J.: OPPNets and rural areas: an opportunistic solution for remote communications. Wirel. Commun. Mobile Comput. (2021). https://doi.org/10.1155/2021/8883501
5. Jesus-Azabal, M., Berrocal, J., Soares, V.N., García-Alonso, J., Galán-Jiménez, J.: A self-sustainable opportunistic solution for emergency detection in ageing people living in rural areas. Wirel. Netw. **29**(5), 2353–2370 (2023)
6. Trotta, A., Andreagiovanni, F.D., Di Felice, M., Natalizio, E., Chowdhury, K.R.: When UAVs ride a bus: towards energy-efficient city-scale video surveillance. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pp. 1043–1051 (2018). https://doi.org/10.1109/INFOCOM.2018.8485863
7. Song, B.D., Park, K., Kim, J.: Persistent UAV delivery logistics: milp formulation and efficient heuristic. Comput. Ind. Eng. **120**, 418–428 (2018). https://doi.org/10.1016/j.cie.2018.05.013
8. Erdelj, M., Natalizio, E., Chowdhury, K.R., Akyildiz, I.F.: Help from the sky: leveraging uavs for disaster management. IEEE Pervasive Comput. **16**(1), 24–32 (2017). https://doi.org/10.1109/MPRV.2017.11

9. Galán-Jiménez, J., Moguel, E., García-Alonso, J., Berrocal, J.: Energy-efficient and solar powered mission planning of UAV swarms to reduce the coverage gap in rural areas: the 3d case. Ad Hoc Netw. **118**, 102517 (2021). https://doi.org/10.1016/j.adhoc.2021.102517
10. Amorosi, L., Chiaraviglio, L., Galán-Jiménez, J.: Optimal energy management of UAV-based cellular networks powered by solar panels and batteries Formulation and solutions. IEEE Access **7**, 53698–53717 (2019)
11. Galán-Jiménez, J., Vegas, A.G., Berrocal, J.: Energy-efficient deployment of iot applications in remote rural areas using UAV networks. In: 2022 14th IFIP Wireless and Mobile Networking Conference (WMNC), pp. 70–74 (2022). https://doi.org/10.23919/WMNC56391.2022.9954292
12. Shin, A., Lim, Y.: Federated-learning-based energy-efficient load balancing for UAV-enabled mec system in vehicular networks. Energies (2023). https://doi.org/10.3390/en16052486
13. Huang, S., Zeng, D., Qu, Z.: Toward performance efficient UAV task scheduling in cloud native edge. In: GLOBECOM 2022 - 2022 IEEE Global Communications Conference, pp. 4517–4522 (2022). https://doi.org/10.1109/GLOBECOM48099.2022.10001252
14. Fu, S., Yin, L., Jiang, C., Jamalipour, A.: An energy-efficient intelligent framework of uav-enhanced vehicular networks: open problems and a case study. IEEE Vehicular Technology Magazine **17**(2), 94–102 (2022). https://doi.org/10.1109/MVT.2022.3157068
15. Van Huynh, D., Do-Duy, T., Nguyen, L.D., Le, M.-T., Vo, N.-S., Duong, T.Q.: Real-time optimised path planning and energy consumption for data collection in UAV-aided intelligent wireless sensing. IEEE Transactions on Industrial Informatics (2021)
16. Ouyang, W., Chen, Z., Wu, J., Yu, G., Zhang, H.: Dynamic task migration combining energy efficiency and load balancing optimization in three-tier UAV-enabled mobile edge computing system. Electronics (2021). https://doi.org/10.3390/electronics10020190
17. Al-Hourani, A., Kandeepan, S., Lardner, S.: Optimal lap altitude for maximum coverage. IEEE Wirel. Commun. Lett. **3**(6), 569–572 (2014)
18. Chiaraviglio, L., Galán-Jiménez, J., Fiore, M., Blefari-Melazzi, N.: Not in my neighborhood: a user equipment perspective of cellular planning under restrictive emf limits. IEEE Access **7**, 6161–6185 (2019). https://doi.org/10.1109/ACCESS.2018.2888916
19. Galán-Jiménez, J., Chiaraviglio, L.: Measuring the impact of icnirp vs. stricter-than-icnirp exposure limits on qos and emf from cellular networks. Comput. Netw. **187**, 107824 (2021). https://doi.org/10.1016/j.comnet.2021.107824
20. Kasun Indrasiri: Microservices in Practice - Key Architectural Concepts of an MSA: https://wso2.com/whitepapers/microservices-in-practice-key-architectural-concepts-of-an-msa/. Accessed 31 May 2023
21. Mishra, S.K., Puthal, D., Rodrigues, J.J.P.C., Sahoo, B., Dutkiewicz, Eryk: Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. IEEE Trans. Ind. Inform. **14**(10), 4497–4506 (2018). https://doi.org/10.1109/TII.2018.2791619
22. Hayyolalam, V., Aloqaily, M., Ozkasap, O., Guizani, M.: Edge intelligence for empowering iot-based healthcare systems. Preprint at arXiv:2103.12144 (2021)
23. Chu, Y., Ho, C., Lee, Y., Li, B.: Development of a solar-powered unmanned aerial vehicle for extended flight endurance. Drones **5**(2), 44 (2021)
24. Limaye, A., Adegbija, T.: A workload characterization for the internet of medical things (iomt). In: 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 302–307 (2017). https://doi.org/10.1109/ISVLSI.2017.60
25. ARM Cortex-A53 MPCore Processor Technical Reference Manual r0p3. https://developer.arm.com/documentation/ddi0500/e/level-1-memory-system/about-the-l1-memory-system. Accessed 31 May 2023
26. Jayasankar, U., Thirumal, V., Ponnurangam, D.: A survey on data compression techniques: from the perspective of data quality, coding schemes, data type and applications. J. King Saud Univ.- Comput. Inform. Sci. **33**(2), 119–140 (2021). https://doi.org/10.1016/j.jksuci.2018.05.006
27. Milosevic, M., Dzhagaryan, A., Jovanov, E., Milenković, A.: An environment for automated power measurements on mobile computing platforms. In: Proceedings of the 51st ACM Southeast Conference. Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2498328.2500064