








# Decentralized Algorithms for Efficient Energy Management over Cloud-Edge Infrastructures

Aristeidis Karras<sup>1</sup><sup>(✉)</sup>, Christos Karras<sup>1</sup>, Ioanna Giannoukou<sup>2</sup>,  
Konstantinos C. Giotopoulos<sup>2</sup>, Dimitrios Tsolis<sup>3</sup>, Ioannis Karydis<sup>4</sup>, and  
Spyros Sioutas<sup>1</sup>

<sup>1</sup> Computer Engineering and Informatics Department, University of Patras, 26504 Patras, Greece

{akarras,c.karras,sioutas}@ceid.upatras.gr

<sup>2</sup> Department of Management Science and Technology, University of Patras, 26334 Patras, Greece

{igian,kgiotop}@upatras.gr

<sup>3</sup> Department of History and Archaeology, University of Patras, 26504 Patras, Greece

dtsolis@upatras.gr

<sup>4</sup> Department of Informatics, Ionian University, 49100 Corfu, Greece

karydis@ionio.gr

**Abstract.** This paper presents an innovative approach to overcoming the limitations of traditional cloud-centric architectures in the evolving Internet of Things (IoT) landscape. We introduce a set of novel decentralized algorithms boosting Mobile Edge Computing (MEC), a paradigm shift towards placing computational resources near data sources, thus boosting real-time processing and energy efficiency. Our approach addresses the challenges of managing distributed Cloud-edge infrastructures in high-mobility environments, such as drone networks. Utilizing the Random Waypoint Model to anticipate device trajectories, our algorithms ensure effective resource allocation, enhanced load balancing, and improved Quality of Service (QoS). An in-depth complexity analysis further improves the scalability and performance of our method, demonstrating their ability to optimize energy efficiency and minimize latency, offering optimized offloading strategies in dynamic IoT environments.

**Keywords:** Decentralized Computing · Cloud Computing · Edge Computing · IoT Systems · Energy Management · High-mobility Environments.

## 1 Introduction

The rapid proliferation of Internet of Things (IoT) technology in recent years, alongside the increasing demand for high-performance, low-latency applications, has placed a significant role on existing cloud-centric infrastructures. IoT devices, commonly referred to as edge devices due to their proximity to data sources, frequently generate massive volumes of data that need to be processed in real-time.

However, the limitive nature of centralized cloud architectures can introduce latency that undermines the real-time processing capabilities and energy efficiency of such systems. Some emerging paradigms, such as the Mobile Edge Computing (MEC), are designed to overcome these constraints by placing computation and storage resources closer to data sources, thereby enabling prompt data processing while conserving energy sources.

This progression towards the network edge indicates a transformative change in infrastructure management, where tasks are no longer exclusively offloaded to a central Cloud. Instead, computational workloads can be distributed across the edge of a network, ensuring the proximity of MEC servers to data sources for swift, efficient computation. By bringing computation closer to end-devices, MEC minimizes transmission delays and reduces the volume of data that must traverse the network, leading to enhanced quality of service (QoS) and improved energy efficiency.

Apart from the potential benefits, the effective management of these distributed, Cloud-edge infrastructures presents a complex set of challenges. In a dynamic environment characterized by high device mobility, such as drones, finding the optimal offloading strategy becomes a challenging problem. Existing offloading methods that assume static device positions may prove inadequate in this new context. The added complexity of drone mobility necessitates algorithms that can efficiently determine the most suitable MEC server for task execution. The challenges are further compounded by the multitude of available MEC servers, which could lead to suboptimal offloading decisions if tasks are blindly assigned to directly linked servers.

The objective of this paper is to delve into these complexities and propose a novel set of decentralized algorithms that optimize energy management across cloud-edge infrastructures with mobile edge devices, like drones and also to assess mobility and task allocation. Our proposed algorithms anticipate device mobility, utilizing the Random Waypoint Model to predict device trajectories and appropriately allocate resources. With the consideration of unique delay restrictions for each task, our approach aims to strike a balance between efficient energy use and minimal latency, thereby improving QoS.

This paper is organized as follows: Section 2 discusses the relevant literature. Section 3.2 presents the architecture of our model and formulates the problem. The proposed algorithms are elaborated in Section 3.3. Section 4 assesses the performance of these algorithms and discusses the results. Finally, Section 5 concludes the paper and explores potential future research directions.

## 2 Background and Related Work

In the dynamic and swiftly-evolving field of cloud computing, and its increasingly significant extensions—edge, fog, and decentralized architectures—several innovative improvements are changing the topic of resource management and performance optimization across a combination of applications. Essential to this transformation, tools, and algorithms such as CloudSim, a toolkit developed

by Calheiros et al. [2], and the novel bio-inspired hybrid algorithm (NBIHA) proposed by Rafique et al [17]. These novel ideas aim to increase resource provisioning and improve energy efficiency and execution times within these different computing environments, thereby establishing new benchmarks for enhancing operational efficiency and performance.

Smart grid systems are also significantly affected by the ground-breaking impact of these cutting-edge developments. Chekired et al.'s decentralized Cloud-SDN architecture, which employs a dynamic pricing model [4], and Zahoor et al.'s cloud-fog-based smart grid model are establishing the way for this technological breakthrough [23,24]. The model proposed by Zahoor effectively combines the ideas of ant colony optimization and artificial bee colony optimization. These revolutionary improvements not only enhance efficient energy management and utilization but additionally optimize resource utilization, even in the most high-demanding cases, demonstrating effectiveness and flexibility in the current energy environment.

Alongside the recent advances in smart grid systems, there is a simultaneous growth of innovation that incorporates the power of cloud computing with the broad Internet of Things (IoT) network. Han et al.'s efficient deep learning framework for intelligent energy management effectively incorporates this synergy [8]. This framework improves smart grid models, such as those presented by [4] and [23,24], by facilitating the demand for energy and response processes efficiently. Pantazoglou et al. have proposed a decentralized, energy-efficient workload management system for enterprise clouds to complement this technological integration [16]. Each node operates independently, simulating the autonomous behavior of ant and bee colonies in Zahoor's model, consequently enhancing the autonomy and efficiency of the system as an entire unit.

Regarding the progress made in combining cloud computing and IoT networks, major improvements are also being made in the fields of edge networks and vehicular edge computing. Liu et al. are on the leading edge with their multi-factor energy-aware resource management system [14], while Wang et al. have developed a cloud-edge collaborative strategy for computation offloading [21]. Alongside addressing increased traffic flow demands and expansive distribution distances, these cutting-edge methodologies enhance vehicular edge computing performance to levels never before achieved.

Machine learning and deep reinforcement learning approaches have become powerful factors in the direction of enhanced energy efficiency and optimized resource utilization. This is demonstrated by Tian et al.'s decentralized collaborative power management system [20]. By applying the power of multi-device knowledge sharing, their system leads to significant energy savings. Alongside this, Jayanetti et al. have utilized deep reinforcement learning to create an innovative scheduling framework, deftly managing complex workflow scheduling problems in edge-cloud environments [9].

As a result of the progress made essential through machine learning and deep reinforcement learning, Rey-Jouanchicot et al. [18] and Blanco et al. [6] emphasize on the essential role that IoT device availability and robust consensus models

have. These components prove to be critical in determining computational capacity and managing resources in applications such as smart buildings or cities. Xiong et al.'s study on blockchain network management provides a major new dimension to the optimization of system performance by delving into this complex and data-rich environment [22]. Ultimately, these innovations represent the rapid development and great possibilities offered by cloud, edge, and fog computing, significantly enhancing the performance of a wide spectrum of applications.

## 2.1 Decentralized Energy Management

The complexities of energy management in cloud-edge infrastructures arise due to the dynamic nature of resources, especially when compared to traditional cloud settings [13]. Edge computing, a decentralized paradigm, capitalizes on resources at the network's edge to facilitate local data processing, making it closer to user-end devices such as smartphones or wearables. Recently, edge, there has been a marked increase in the application of machine learning (ML) at this network edge. This trend mainly seeks to enhance computational services, especially focusing on reduced latency, energy conservation, and resource optimization [1].

In the exploration of energy-efficient resource allocation, adapting distributed machine learning (ML) algorithms for execution at the edge is crucial. This strategic adaptation promotes synergy with cloud systems, aiming to achieve reduced latency, enhanced energy efficiency, safeguarded user privacy, and improved system scalability [15]. Recent technological advancements encompass a deep reinforcement learning-based mechanism, specifically designed for cloud-edge collaborative offloading, addressing the dynamic requirements of industrial networks [5]. Additionally, for advanced energy management in smart grids, a privacy-focused average consensus algorithm has been introduced, seamlessly integrating the benefits of both cloud and edge computing [7].

In summary, achieving energy efficiency in cloud-edge infrastructures primarily requires adapting machine learning algorithms for edge environments, utilizing distributed learning techniques, and designing privacy-centric methodologies to ensure secure and effective resource allocation.

## 2.2 Mobile Edge Computing vs Traditional Cloud-Centric Architectures

Mobile Edge Computing (MEC) is a paradigm shift towards placing computational resources near data sources, thus boosting real-time processing and energy efficiency. MEC is a distributed computing architecture that extends cloud computing capabilities to the edge of the network, closer to the end-users and data sources. This tension towards the network edge indicates a transformative change in infrastructure management, where tasks are no longer exclusively offloaded to a central Cloud. Instead, computational workloads can be distributed across the edge of a network, ensuring the proximity of MEC servers to data sources for swift, efficient computation. By making the computation closer to end devices, MEC minimizes transmission delays and reduces the volume of data that must

traverse the network, leading to enhanced quality of service (QoS) and improved energy efficiency. In contrast, traditional cloud-centric architectures rely on centralized data centers to process and store data, which can result in higher latency and increased network traffic.

### 2.3 Energy-efficient resource allocation in Cloud-Edge Infrastructures.

The escalating demand for computing capabilities coupled with increasing energy consumption of data centers has emphasized the significance of energy-efficient resource allocation in cloud-edge infrastructures [19,3]. Integrating renewable energy into data centers offers the potential to reduce their energy use and carbon footprints [12]. However, the natural variability of renewable sources often results in under-utilization. To counter this, research has focused on two primary strategies: energy storage and opportunistic scheduling [12].

In the context of mini data centers, combined optimization of virtual machines (VMs) and energy resources has shown to reduce grid electricity usage by 22%, as compared to solely focusing on VM allocation [19]. This reduction further extends to 28.5% when considering less energy-efficient servers [19]. In distributed cloud-edge systems, the challenge of joint workload distribution and computational resource adjustment has been approached using the Dynamic Voltage and Frequency Scaling (DVFS) method. By dynamically adjusting VM computation frequencies based on demand, this method offers energy conservation benefits [25].

Regarding UAV-enabled secure edge-cloud computing systems, strategies for efficient resource allocation and computation offloading have been identified as key to reducing energy consumption [11]. This problem is approached by segmenting it into resource allocation, task distribution, and computation offloading. Systematic solutions for each segment have been proposed to ensure energy-efficient resource allocation and offloading [11].

In conclusion, achieving energy efficiency in cloud-edge infrastructures demands a comprehensive approach. This includes the integration of renewable energy, strategic scheduling, optimized VM and energy allocation, and advanced computation offloading methods. These collective efforts can lead to considerable reductions in energy consumption and carbon output, addressing the growing need for computational resources.

### 2.4 Comparison Analysis of Algorithms

In recent years, the necessity for optimized resource allocation in computing systems has yielded several algorithms, with OptiMEC leading the initial charge. Introduced in our foundational work [10], OptiMEC sought to efficiently allocate resources by minimizing energy consumption. Building upon the groundwork laid by OptiMEC, this paper presents advancements in the form of three novel methods: EffiMEC, E-OptiMEC, and a dedicated Load-Balancing approach.

EffiMEC shifts its primary focus to maximizing efficiency, while E-OptiMEC combines elements from both OptiMEC and EffiMEC, targeting energy-efficient solutions with an emphasis on drones. Lastly, our Load-Balancing method introduces a dynamic way of distributing workloads, thereby reducing latency and further promoting energy efficiency.

Table 1 below provides a comparative analysis of these methods, shedding light on their objectives, focal points, and underlying mechanisms.

**Table 1.** Comparative Analysis of Algorithms

| Feature          | Aspect              | Algorithm                  |                        |                                   |                                    |
|------------------|---------------------|----------------------------|------------------------|-----------------------------------|------------------------------------|
|                  |                     | OptiMEC                    | EffiMEC                | E-OptiMEC                         | Load-Balancing                     |
| <b>Objective</b> | Minimize Energy     | ✓                          | -                      | ✓                                 | -                                  |
|                  | Maximize Efficiency | -                          | ✓                      | -                                 | -                                  |
| <b>Focus</b>     | Mobility            | User                       | Drone                  | Drone                             | UAV                                |
|                  | Load Distribution   | -                          | -                      | -                                 | ✓                                  |
| <b>Input</b>     | Task                | Task, User<br>Mob., Deadl. | Task, Server<br>Effic. | Task, Drone<br>Mob., Deadl.       | Task, UAV<br>Mob., Deadl.          |
| <b>Output</b>    | Type                | Workload<br>Distrib.       | Server<br>Selection    | Drone-based<br>Distrib.           | Balancing<br>Plan                  |
| <b>Mechanism</b> | Action              | Min. Energy<br>for Tasks   | Max.<br>Efficiency     | Identify Low<br>Energy<br>Servers | Balance<br>Workload<br>and Latency |

### Comparative Overview

- **OptiMEC** [10]: Focused on scenarios involving mobile users, OptiMEC diligently minimizes energy use, accounting for user mobility and task deadlines through its predictive architecture.
- **EffiMEC**: Developed as an alternative for UAVs, EffiMEC transitions from a sole focus on energy to prioritize computational efficiency, adjusting structural considerations to suit drone operations.
- **E-OptiMEC**: A drone-adapted variant, E-OptiMEC, maintains an energy optimization focus, adjusted for drone-specific contexts.
- **Load-Balancing**: The Load-Balancing method is a comprehensive strategy accentuating real-time task distribution while ensuring adherence to energy constraints and task deadlines.

### Structural and Architectural Variances with OptiMEC

- **Energy versus Efficiency**: While OptiMEC seeks to optimize energy consumption, EffiMEC targets optimal computational efficiency.
- **Mobility Consideration**: OptiMEC’s architecture is devised for mobile user scenarios, a stark contrast to E-OptiMEC and Load-Balancing, both of which cater to UAV operations.

- Task Management Mechanisms: Load-Balancing introduces a more refined task management framework, contrasting with the more general approach in OptiMEC.

**Design Distinctions** The choice between EffiMEC, E-OptiMEC, or the Load-Balancing algorithm should align with specific operational priorities and contexts:

- EffiMEC is optimal in environments where computational efficiency takes precedence over energy conservation.
- E-OptiMEC and Load-Balancing are best suited for scenarios that demand a comprehensive strategy, especially when contending with dynamic task necessities and constrained energy supplies.
- OptiMEC remains the preferred choice for environments primarily driven by the objective of energy conservation, particularly in mobile user scenarios.

**Final Analysis of Proposed Algorithms: EffiMEC and E-OptiMEC** The algorithms introduced in this paper, EffiMEC and E-OptiMEC, have been compared with two alternative methods: Random Allocation and Load-Balancing. Both methods are explained in depth in our previous work [10]. Analytical results demonstrate that EffiMEC emerges as the most efficient, registering the least average energy consumption, with E-OptiMEC showcasing a comparable performance.

In its operational mechanics, the E-OptiMEC heuristic assigns each task to a MEC server, ensuring the task deadline is achieved while optimizing energy consumption. At every time interval, for each drone, the heuristic evaluates accessible MEC servers, gravitating towards the one that minimizes total energy expended during task execution. The end product of this algorithm is a server assignment aimed at attenuating the transmission energy indispensable for task distribution. When contrasted with the Random-Allocation method, OptiMEC’s energy efficiency surpasses it by 10.42%.

Load-Balancing serves as a standard reference for this analysis. The data affirms that both EffiMEC and E-OptiMEC display superior performance over Load-Balancing.

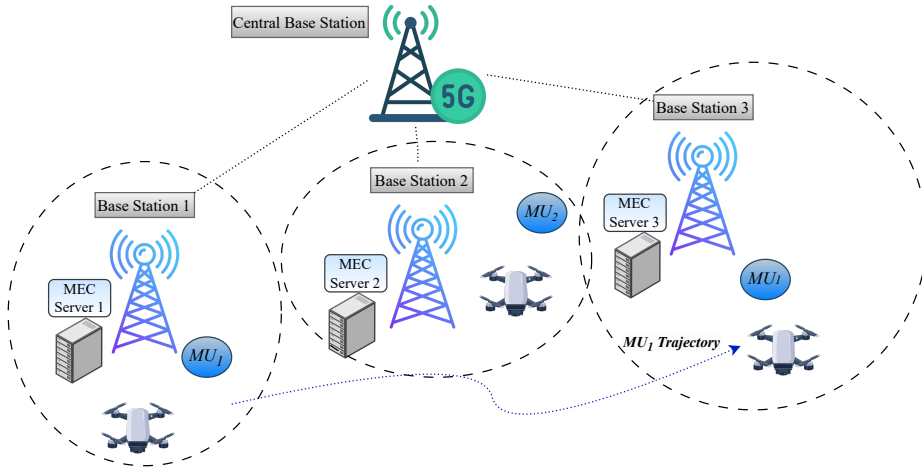
### 3 Methodology

#### 3.1 Problem Formulation

A Mobile Edge Network (MEN) is structured with multiple Base Stations (BS). Each BS is fortified with a MEC server, tailored for energy-efficient operations as per algorithms like OptiMEC and EffiMEC which are further analysed below. These servers are adept at receiving, processing, and relaying computational tasks offloaded by users within their signal domains. Notably, BSs are interlinked

via a Central Base Station (CBS), ensuring efficient load-balancing and task distribution.

Mobile users (MUs) or Mobile Devices (MDs), in their quest for reduced latency and energy conservation, as emphasized by the E-OptiMEC framework, can offload tasks to MEC servers. Given the inherent mobility, users and devices might transition beyond the range of an initial BS. As illustrated in Fig. 1, after MU1 offloads task T1 to BS1, it moves out of its range. Here, leveraging the adaptive strategies of algorithms, the MEC at BS1 can forward the task to the MEC at BS2, aligning with the user's current location. This ensures any MEC within the user's trajectory remains poised to execute the offloaded task, optimizing both energy consumption and computational efficiency.



**Fig. 1.** Mobile Edge Computing Network.

The problem can be articulated as a mathematical optimization challenge with the objective of minimizing the total energy consumption for all mobile devices represented by the set  $N$ . The aim is to determine the optimal MEC server from a set of accessible servers, denoted for each trajectory of a device as  $S_i$ , that meets each stipulated task deadline. A pivotal hurdle lies in detecting available MEC servers when a user relocates and in orchestrating task computation at a designated server.

Let us denote  $N = \{1, 2, \dots, i, \dots, N\}$  as the mobile devices traversing the area overseen by MEN. Every user possesses a task  $T_i$ , characterized as a triplet  $(s_i, c_i, t_{\max})$ , where  $s_i$  represents the size of the computational task,  $c_i$  is the essential computation resource measured in cycles, and  $t_{\max}$  denotes the task's deadline. Furthermore, there are  $M$  base stations in total, with each having a signal range  $r_j$  and the corresponding MEC boasting a computation capacity of  $a_j$ .



### 3.2 Proposed Decentralized Energy-Efficient Algorithm

Given the complexity of the problem formulated in the previous sections, a novel decentralized algorithm is proposed to solve the problem in a scalable manner while preserving energy efficiency. This algorithm, referred to as the Decentralized Energy-efficient Offloading and Resource Allocation (EffiMEC), is designed to handle drone mobility and dynamically offload tasks to different edge servers following the trajectory of mobile drones. The EffiMEC approach is inspired by swarm intelligence, particularly ant colony optimization, with adaptations to suit the MEC context.

### 3.3 Algorithm Description

Each Mobile Device (MD) in the network runs an instance of the EffiMEC algorithm, enabling a fully decentralized system. The algorithm works as follows:

1. Each MD  $i$  initiates the algorithm by considering its current location and available MEC servers within its range. This set of servers is denoted by  $S_i^0$ .
2. The MD calculates the energy cost  $E_{i,j}$  and execution time  $t_{i,j}$  for each available MEC server  $j$  in  $S_i^0$ . These values are used to evaluate the suitability of each server.
3. For each server  $j$  in  $S_i^0$ , MD calculates the quality of the server,  $Q_{i,j}$ , as follows:

$$Q_{i,j} = \frac{1}{E_{i,j}} - \lambda \cdot t_{i,j} \quad (1)$$

where  $\lambda$  is a tunable parameter representing the trade-off between energy conservation and time efficiency. A higher  $\lambda$  gives more importance to time efficiency, while a lower  $\lambda$  gives more importance to energy conservation.

4. The MD then probabilistically selects the next MEC server to offload the task based on the quality of the servers. The probability  $P_{i,j}$  of selecting server  $j$  is calculated as follows:

$$P_{i,j} = \frac{Q_{i,j}}{\sum_{k \in S_i^0} Q_{i,k}} \quad (2)$$

5. After offloading the task to the selected MEC server, the MD moves to the next location, and the set of available MEC servers  $S_i^t$  gets updated. The above steps are repeated until the task is fully executed or the task deadline is reached.

### 3.4 Proposed Heuristic: Efficiency Maximizing Algorithm

In Algorithm 1, a series of symbols are utilized to define, calculate, and manipulate various parameters related to the task allocation problem, drones, servers, and other computational elements. A comprehensive list of these symbols and their corresponding definitions is presented in Table 2. These symbols represent

various entities and metrics such as the drones, servers, task-related parameters, time intervals, and efficiencies, which are critical for the functioning and explanation of the EffiMEC algorithm. The specific use and context of each symbol are based upon the operations and calculations performed within the algorithm, and they collectively serve to establish a mathematical and logical framework to elucidate the heuristic's inner workings and methodologies. Understanding these symbols and their respective roles within the algorithm is vital for comprehending the mechanics, analyses, and results of the proposed heuristic approach.

**Table 2.** Symbol Definitions for EffiMEC Algorithm

| Symbol                 | Definition   |
|------------------------|--|
| $P1$                   | The task allocation problem  |
| $i$                    | Drone identifier   |
| $t$                    | Time step or interval  |
| $mt_i$                 | Final time step for drone $i$  |
| $N$                    | Set of mobile drones   |
| $M$                    | Set of base stations   |
| $S_i^t$                | Available MEC servers for drone $i$ at time $t$                      |
| $T_i$                  | Task assigned to drone $i$   |
| $td_i$                 | Deadline for task $T_i$  |
| $t_j^e$                | Time required for task execution on server $j$                       |
| $t_{i,j}^u$            | Time to upload task from drone $i$ to server $j$                     |
| $t_{i,j}^e$            | Execution time of task on server $j$ from drone $i$                  |
| $eff_j$                | Efficiency of server $j$   |
| $c_{i,j}$              | Computational capacity of server $j$ for drone $i$                   |
| $e_{i,j}$              | Energy consumed for task execution on server $j$ from drone $i$      |
| $t_{j0\dots t-1}^{tr}$ | Transmission time from server $j$ at previous time steps             |
| $t_{jt-1,j}^{tr}$      | Transmission time from server $j$ at time $t-1$ to $j$ at time $t$   |
| $e_{j0\dots t-1}^{tr}$ | Transmission energy from server $j$ at previous time steps           |
| $e_{jt-1,j}^{tr}$      | Transmission energy from server $j$ at time $t-1$ to $j$ at time $t$ |
| $y_i$                  | Assigned server for drone $i$  |

The task allocation problem  $P1$  is a combinatorial optimization problem, classified as NP-hard. To address this issue, we propose an efficiency-maximizing heuristic called "EffiMEC". Similar to the OptiMEC algorithm, each drone  $i$  offloads a task  $T_i$  with details about the task size, required computational resources, and deadline.

For each drone  $i \in N$  and at each time interval  $t \in [0, mt_i]$ , the CBS identifies available MEC servers, denoted as  $S_i^t$ . The objective is to find an optimal server from  $S_i^t$  for computation and a suitable server for transmission of the task to another server in the next time step region. The time required for uploading, transmission and computation is calculated and compared against the

task deadline at each time step. The efficiency of each available server in  $S_i^t$  is computed by considering the computational capacity and the energy consumed during transmission and execution up to time step  $t - 1$ .

---

**Algorithm 1** EffiMEC Algorithm
 

---

**Input:** mobile drones  $N$ , base stations  $M$ 
**Output:**  $y_i, \forall i \in M$ 

```

1: for each drone  $i \in N$  do
2:    $S_i^t \leftarrow$  get available MEC servers at final time step ( $mt_i$ )
3:   for each server  $j \in S_i^t$  do
4:     if  $t == 0$  then
5:        $t_j^e = t_{i,j}^u + t_{i,j}^e$ 
6:       if  $t_j^e \leq td_i$  then
7:          $eff_j = (c_{i,j} - e_{i,j})/c_{i,j}$ 
8:       else
9:          $eff_j \leftarrow 0$ 
10:      end if
11:     else
12:        $t_j^e = t^{tr} j 0 \dots t - 1 + t^{tr} j t - 1, j + t_{i,j}^e$ 
13:       if  $t_j^e \leq td_i$  then
14:          $eff_j = (c_{i,j} - e^{tr} j 0 \dots t - 1 - e^{tr} j t - 1, j - e_{i,j})/c_{i,j}$ 
15:       else
16:          $eff_j \leftarrow 0$ 
17:       end if
18:     end if
19:   end for
20:    $j_*^t = \text{argmax}_j(eff_j)$ 
21: end for
22: return  $y_i, \forall i \in M$ 

```

---

This procedure is reiterated for all time intervals. The server with the highest efficiency is assigned to the task  $T_i$  for execution. The efficiency of a server at time step  $t$  can be calculated by subtracting the energy consumed for transmission up to time step  $t - 1$  and the energy required to offload the task from the server's location to the next time step's locality from the server's computational capacity. The algorithm finally outputs  $y_i \forall i \in N$ , the server assignment that maximizes efficiency while satisfying the delay constraints.

In addition, we propose a resource-allocation algorithm. This heuristic ensures the efficient use of available resources by balancing the computation load among servers. This algorithm is provided in Algorithm 3.

### 3.5 Complexity Analysis of the EffiMEC Algorithm

The time complexity of the EffiMEC algorithm arises from its nested loop structure. The outer loop traverses all  $N$  mobile drones, while the inner loop iterates over  $M$  MEC servers available to each drone. As such, the worst-case scenario presents a time complexity of  $O(NM)$ . The operations within these loops, including computations, condition checking, and assignments, possess a constant time complexity of  $O(1)$ . Therefore, it does not affect the overall time complexity. It is important to note, the complexity could increase if the invoked functions, such as getting available MEC servers at the final time step have higher time complexities. Regarding space complexity, the EffiMEC algorithm appears to use a constant amount of space ( $O(1)$ ), not necessitating data structures that scale with the size of the input. This analysis considers the worst-case scenario. Depending on server distribution and specific implementation, the actual performance may vary.

### 3.6 Proposed Scheduling Framework

Our problem  $P1$  has been identified as a constraint satisfaction problem and is of NP-complete nature. To solve it, we put forward a scheduling heuristic. The heuristic works as follows: Every drone  $i$  offloads the task  $T_i$  along with necessary information such as the size of the computational task, required computation resources (expressed in cycles), and task deadline. The Central Base Station (CBS) allocates each task to a Multi-access Edge Computing (MEC) server that fulfils the task deadline and ensures minimum energy consumption. We call this mechanism the Energy-efficient Optimal Multi-Access Edge Computing (E-OptiMEC) algorithm, and its procedure is detailed in Algorithm 2.

At each time step  $t \in [0, mt_i]$  for every drone  $i \in N$ , the heuristic identifies the available MEC servers, denoted as  $S_i^t$ . Among the available servers, the goal is to identify two servers - one for task execution, and one for task transmission, to offload the task to a server in the subsequent time step's location. Given that the task assignment must adhere to the delay constraint, the time required for uploading, transmission, and execution is calculated and checked against the delay constraint at each time step. The algorithm also computes the energy consumed due to transmission and execution for each available server.

This process is replicated across all temporal instances, whereby the server demonstrating the lowest energy expenditure is elected for executing task  $T_i$ . The transmission energy at a given timestamp  $t$  can be derived by combining the cumulative transmission energy until timestamp  $t-1$  and the energy mandated to transfer the task from the incumbent server's location to the drone's position at the ensuing timestamp  $t+1$ . In the final analysis, the algorithm yields  $x_i \forall i \in N$ , specifying the server that optimizes energy utilization while adhering to the latency stipulations of task  $T_i$ .

---

**Algorithm 2** E-OptiMEC Algorithm

---

**Input:** mobile drones  $N$ , base stations  $M$ **Output:**  $x_i, \forall i \in M$ 

- 1: **for each** drone  $i \in N$  **do**
  - 2:    $S_i^t \leftarrow$  identify available MEC servers at the final time step ( $mt_i$ )
  - 3:   **for each** server  $j \in S_i^t$  **do**
  - 4:     Calculate time and energy for task execution and transmission
  - 5:     Identify suitable servers for task execution and transmission based on least energy consumption and deadline satisfaction
  - 6:   **end for**
  - 7:   Select the server with the minimum total energy consumption
  - 8: **end for**
  - 9: **return**  $x_i, \forall i \in M$
- 

**3.7 Complexity Analysis of the E-OptiMEC Algorithm**

The time complexity of the E-OptiMEC algorithm derives primarily from its nested loop construction. The outer loop iterates over each of the  $N$  mobile drones, while the inner loop traverses  $M$  MEC servers accessible to a given drone. Consequently, the worst-case time complexity is  $O(NM)$ . Within these loops, operations such as calculations, server identification based on energy consumption and deadline satisfaction, and server selection all carry a constant time complexity,  $O(1)$ . Thus, they do not impact the overall time complexity.

However, the total time complexity could be greater if the internal functions, like identifying available MEC servers at the final time step, have higher time complexities. As for the space complexity, the E-OptiMEC algorithm uses a constant amount of space,  $O(1)$ , since it does not require data structures scaling with the size of the input. This analysis represents a worst-case scenario. The actual performance may be more efficient, contingent upon the specific server distribution and implementation details.

Moreover, we offer an example of a load-balancing heuristic algorithm designed to balance each mobile drone's energy consumption across available servers. This algorithm is depicted in Algorithm 3.

Algorithm 3 makes use of the energy consumption information of each mobile drone among the available servers. It selects the server with the lowest energy consumption for task assignment. If the energy balance for mobile drones falls below the threshold  $e_t$ , the algorithm includes a step to postpone tasks or reduce task complexity. Moreover, the algorithm updates each drone's energy balance after each workload distribution, facilitating more precise load-balancing decisions.

---

**Algorithm 3** Power-Efficient Workload Distribution Using E-OptiMEC
 

---

**Input:** UAVs  $N$ , base stations  $M$ , energy cap  $e_t$ 
**Output:**  $x_i, \forall i \in N$ 

```

1: for each UAV  $i \in N$  do
2:   for each timestep  $t \in [0, mt_i]$  do
3:      $S_i^t$  locates reachable MEC servers at timestep  $t$ 
4:     Evaluate energy usage for each UAV  $i$ 
5:     Pinpoint server with smallest energy drain ( $j_t$ )
6:     Amend energy balance record for UAV  $i$ 
7:     if UAV  $i$  energy balance  $< e_t$  then
8:       defer task or diminish task sophistication
9:     else
10:      consign the task to server  $j_t$ 
11:    end if
12:    Renew the transmission energy ( $e_t r$ ) and transmission interval ( $t_t r$ )
13:  end for
14: end for
15: return  $x_i, \forall i \in N$ 

```

---

### 3.8 Complexity Analysis of the Energy-Efficient Load Balancing Based on E-OptiMEC Algorithm

The Energy-Efficient Load Balancing Based on E-OptiMEC algorithm introduces an additional layer of complexity through its use of two nested loops. The outer loop iterates over the  $N$  mobile drones, while the inner loop goes through each timestep  $t$  within the range  $[0, mt_i]$  for each drone  $i$ .

Without specific knowledge of the variable  $mt_i$ , the worst-case scenario would suggest it's a value of  $T$  where  $T$  is the maximum number of timesteps. Therefore, in the worst case, the inner loop would run  $T$  times for each drone. This results in a worst-case time complexity of  $O(NT)$  for each drone, and with  $M$  base stations, it becomes  $O(NMT)$ , given that  $M$  base stations are checked in each timestep for each drone. The operations within these loops have a constant time complexity of  $O(1)$ . However, as before, the overall time complexity may be higher if the internal functions like determine available MEC servers at timestep  $t$  have time complexities exceeding  $O(1)$ .

As for the space complexity, the algorithm appears to use a constant amount of space ( $O(1)$ ), as it doesn't employ data structures that grow with the input size. Note that this analysis is for the worst-case scenario. The actual performance may be more efficient depending on specific server distribution, the distribution of timesteps per drone, and implementation details.

The prior discourse outlined a load-balancing mechanism among mobile drones in a network, the advantages of which can be enumerated as follows:

1. **Energy efficiency:** The proposed method significantly enhances energy efficiency within a mobile edge computing network. This is achieved by of-

flooding computational tasks to proximate servers, which reduces the energy consumption of mobile devices. Therefore, by regulating the energy expenditure among mobile drones, the cumulative energy consumption of the system can be reduced, fostering a more energy-efficient network.

2. **Equitable workload distribution:** The suggested algorithm ensures a fair distribution of workload among the available servers. This equitable distribution forestalls the overloading of certain servers while others remain underutilized, leading to enhanced resource utilization and improved energy efficiency.

To elucidate the proposed load-balancing algorithm, let's contemplate a hypothetical scenario. Drone 1 offloads tasks to MEC servers  $BS_1$  and  $BS_2$  at timestep 0, as displayed in Table 3. Here,  $BS_2$  is selected for task execution due to its lesser energy consumption post uploading and execution. Concurrently,  $BS_1$  is employed to offload the task to MEC servers in the subsequent timestep due to its lower energy consumption for transmission (refer to Table 4).

**Table 3.** E-OptiMEC at timestep 1

| Execution                      | Transmission              |
|--------------------------------|---------------------------|
| $drone1 \rightarrow BS_1$      | $BS_1 \rightarrow drone1$ |
| $t_1^u + t_1^{ex} = 0.6 (< 1)$ | $t_1^u + t_1^{tr} = 0.3$  |
| $e_1^u + e_1^{ex} = 0.9$       | $e_1^u + e_1^{tr} = 0.6$  |
| $drone1 \rightarrow BS_2$      | $BS_2 \rightarrow drone1$ |
| $t_1^u + t_1^{ex} = 0.7 (< 1)$ | $t_1^u + t_1^{tr} = 0.4$  |
| $e_1^u + e_1^{ex} = 0.8$       | $e_1^u + e_1^{tr} = 0.7$  |

**Table 4.** E-OptiMEC at timestep 2

| Execution                                  | Transmission                        |
|--|-------------------------------------|
| $drone1 \rightarrow BS_1 \rightarrow BS_3$ | $BS_3 \rightarrow drone1$           |
| $t_1^u + t_1^{tr} + t_1^{ex} = 0.9 (< 1)$  | $t_1^u + t_1^{tr} + t_1^{tr} = 0.5$ |
| $e_1^u + e_1^{tr} + e_1^{ex} = 0.7$        | $e_1^u + e_1^{tr} + e_1^{tr} = 0.9$ |
| $drone1 \rightarrow BS_1 \rightarrow BS_4$ | $BS_4 \rightarrow drone1$           |
| $t_1^u + t_1^{tr} + t_1^{ex} = 0.7 (< 1)$  | $t_1^u + t_1^{tr} + t_1^{tr} = 0.6$ |
| $e_1^u + e_1^{tr} + e_1^{ex} = 0.8$        | $e_1^u + e_1^{tr} + e_1^{tr} = 0.8$ |

In the subsequent timestep, the MEC servers at disposal are  $BS_3$  and  $BS_4$ .  $BS_3$  is elected for execution as it consumes lesser energy post uploading and execution. On the other hand,  $BS_4$  is chosen to offload the task to the accessible MEC servers owing to its lower transmission energy requirement. At the final timestep, the available MEC servers are  $BS_5$  and  $BS_6$ .  $BS_5$  is ultimately selected for execution due to its lower energy consumption post uploading and execution.

**Table 5.** E-OptiMEC Execution Phase

| Execution Phase   |
|---|
| $drone1 \rightarrow BS_1 \rightarrow BS_4 \rightarrow BS_5$ |
| $t_1^u + t_1^{tr} + t_1^{tr} + t_1^{ex} = 0.7 (< 1)$        |
| $e_1^u + e_1^{tr} + e_1^{tr} + e_1^{ex} = 1.2$              |
| $drone1 \rightarrow BS_1 \rightarrow BS_4 \rightarrow BS_6$ |
| $t_1^u + t_1^{tr} + t_1^{tr} + t_1^{ex} = 0.6 (< 1)$        |
| $e_1^u + e_1^{tr} + e_1^{tr} + e_1^{ex} = 1.3$              |

Subsequently, as this is the final timestep, there are no further transmissions from  $BS_5$  or  $BS_6$ . The CBS will then delegate the task to either  $BS_2$ ,  $BS_3$ , or  $BS_5$ , contingent on the server that consumes the least execution energy.

The simulation parameters are as follows: the coverage radius of the base station ranges from 70 to 100 meters; the CPU capacity of the MEC is within the range of 7 to 20 GHz; the computation power varies between 3 to 5 watts; the transmission power lies within 0.1 to 1 watt; the channel bandwidth is set at 1 MHz; the mobility speed of the drone is between 1 to 3 Km/h; the task requirement as per CPU capacity ranges from 1 to 10 GHz; the input data size is between 1 to 3 MB; and the task deadline constraint is within 0.1 to 1 second.

For empirical analysis, the proposed algorithm is evaluated in comparison to the random waypoint model and mobile execution (which refers to executing a task on a mobile device instead of a remote server). This comparative analysis provides a robust performance assessment, underscoring the merits of the proposed algorithm.

## 4 Experimental Results

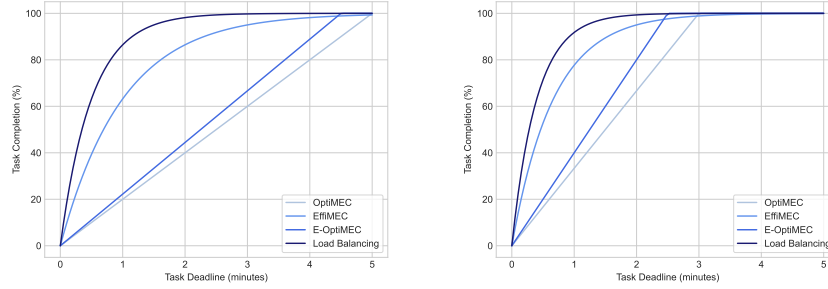
In this section, we evaluate the efficacy of our suggested algorithms in accomplishing minimal-energy task allocation, taking into account the mobility of devices. Moreover, we carry out trials with diverse drone devices, spanning a range of 1 to 10 units. In terms of mobility, we employ data sourced from 57 base stations situated in the central region of London, UK. Moreover, we assess the performance of our proposed algorithms through rigorous experimentation. The primary objectives of these evaluations are twofold as shown in Table 6.

**Table 6.** Evaluation Metrics Description.

| Metric                        | Description  |
|-------------------------------|--|
| Task Completion Efficacy      | Evaluates task completion rates in relation to task deadlines for different drone counts.                                  |
| Energy Consumption Efficiency | Assesses energy usage concerning drone count and task input size. Lower values indicate efficient, sustainable operations. |

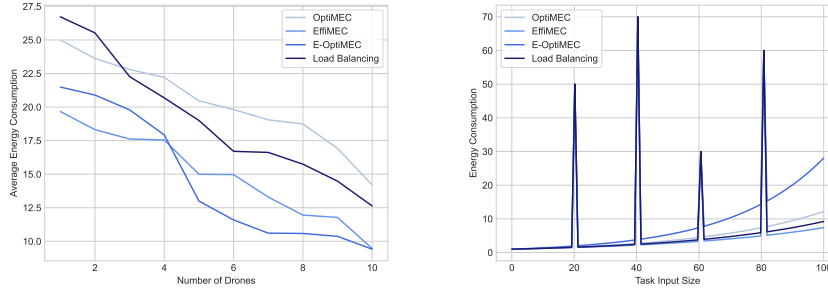
As depicted in Figure 2, it is observed that the Task completion percentage surpasses that of the other three methods when varying Task deadline constraints are applied. The load balancing technique consistently outperforms the other three methods, with EffiMec following closely as the second most effective solution. OptiMEC displays a significant improvement, being 12.56% higher than Random-Allocation and a substantial 29.10% higher than Local-Execution on average. Additionally, a noteworthy observation is the proportional relationship between the Deadline constraint and the Task completion percentage; as





**Fig. 2.** Percentage of Task Completion in correlation to Task Deadline for (a) a single drone and (b) ten drones.

the Deadline constraint is increased, the Task completion percentage likewise sees an increase.



**Fig. 3.** (a) Average Energy Utilization in correlation with Drone Count and (b) Correlation of Task Input Dimension with Energy Expenditure.

Fig. 3 illustrates the fluctuations in energy consumption with varying task input sizes. It becomes abundantly clear that as the task input size escalates, there is a commensurate increase in energy consumption, attributable to the amplified transmission energy required for task distribution. OptiMEC’s energy consumption is more efficient by 10.42% compared to the Random-Allocation method. However, EffiMEC provides an even more efficient solution, boasting the lowest average energy usage, while E-OptiMEC exhibits similar performance. Lastly, within the dimension of Task input size, we notice some spikes coinciding with task arrivals, but the E-OptiMEC technique exhibits superior resilience to these energy fluctuations.

## 5 Conclusions and Future Work

In this work, two novel heuristics are proposed to solve an NP-hard task allocation problem in a Mobile Edge Computing (MEC) context. The first one, EffiMEC, is an efficiency-maximizing algorithm designed to maximize the computational efficiency of servers in the MEC network, while the second one, E-OptiMEC, is an energy-optimizing heuristic aimed at minimizing energy consumption during task execution and transmission. The proposed heuristics operate within a constraint satisfaction framework, wherein tasks from mobile drones are allocated to suitable MEC servers based on energy consumption and computational efficiency while satisfying task deadlines.

EffiMEC algorithm, whose worst-case time complexity is  $O(NM)$ , operates by selecting, for each drone at each time interval, the server with the highest efficiency. Efficiency here is determined by computational capacity minus energy consumed for transmission up to the previous time step and the energy required for offloading the task to the next time step. The output of the algorithm is the server that maximizes efficiency while satisfying the delay constraints.

On the other hand, the E-OptiMEC heuristic operates by allocating each task to a MEC server that fulfils the task deadline and ensures minimum energy consumption. At each time step, for each drone, the heuristic identifies available MEC servers and chooses the one with minimum total energy consumption for task execution. The output of this algorithm is the server assignment that minimizes energy consumption while meeting delay constraints. These two heuristics, based on their designs, could provide more efficient and energy-saving solutions for task allocation problems in MEC networks. However, they are heuristic solutions, and there's no guarantee they'll always achieve the optimal solution. Further, the actual performance of these algorithms may vary depending on server distribution and specific implementation.

As future work, it could be beneficial to conduct comprehensive performance analysis and comparisons of these heuristics in different scenarios or under varying constraints. For instance, scenarios where energy availability or computational capacity is extremely limited, or where task deadlines are particularly stringent. Moreover, while these heuristics are promising, it might be interesting to explore potential enhancements to these algorithms, such as incorporating machine learning techniques for dynamic adaptation, considering other performance metrics like task failure rate, or addressing additional real-world considerations like network congestion and server failure. Finally, the scalability of these heuristics could be evaluated. Given that their complexity scales linearly with the number of drones and servers, understanding how these heuristics perform under heavy network loads could provide valuable insights into their practical applicability in large-scale MEC deployments.

## Acknowledgements

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Pro-

gram Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code: T2EΔK-00127).

## References

1. Angel, N.A., Ravindran, D., Vincent, P.D.R., Srinivasan, K., Hu, Y.C.: Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies. *Sensors* **22**(1), 196 (2021)
2. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* **41**(1), 23–50 (2011)
3. Chauhan, N., Kaur, N., Saini, K.S.: Energy efficient resource allocation in cloud data center: A comparative analysis. In: 2022 International Conference on Computational Modelling, Simulation and Optimization (ICCMO). pp. 201–206 (2022). <https://doi.org/10.1109/ICCMO58359.2022.00049>
4. Chekired, D.A., Khoukhi, L., Mouftah, H.T.: Decentralized cloud-sdn architecture in smart grid: A dynamic pricing model. *IEEE Transactions on Industrial Informatics* **14**(3), 1220–1231 (2018). <https://doi.org/10.1109/TH.2017.2742147>
5. Chen, S., Chen, J., Miao, Y., Wang, Q., Zhao, C.: Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks. *IEEE Transactions on Signal and Information Processing over Networks* **8**, 364–375 (2022). <https://doi.org/10.1109/TSIPN.2022.3171336>
6. Fernandez Blanco, D., Le Mouel, F., Lin, T., Ponge, J.: An energy-efficient faas edge computing platform over iot nodes: Focus on consensus algorithm. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. pp. 661–670 (2023)
7. Fu, W., Wan, Y., Qin, J., Kang, Y., Li, L.: Privacy-preserving optimal energy management for smart grid with cloud-edge computing. *IEEE Transactions on Industrial Informatics* **18**(6), 4029–4038 (2022). <https://doi.org/10.1109/TH.2021.3114513>
8. Han, T., Muhammad, K., Hussain, T., Lloret, J., Baik, S.W.: An efficient deep learning framework for intelligent energy management in iot networks. *IEEE Internet of Things Journal* **8**(5), 3170–3179 (2020)
9. Jayanetti, A., Halgamuge, S., Buyya, R.: Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge-cloud computing environments. *Future Generation Computer Systems* **137**, 14–30 (2022)
10. Karras, A., Karras, C., Giannaros, A., Tsois, D., Sioutas, S.: Mobility-aware workload distribution and task allocation for mobile edge computing networks. In: International Conference on Advances in Computing Research. pp. 395–407. Springer (2023)
11. Khan, U.A., Khalid, W., Saifullah, S.: Energy efficient resource allocation and computation offloading strategy in a uav-enabled secure edge-cloud computing system. In: 2020 IEEE International Conference on Smart Internet of Things (SmartIoT). pp. 58–63 (2020). <https://doi.org/10.1109/SmartIoT49966.2020.00018>
12. Li, Y.: Resource allocation in a Cloud partially powered by renewable energy sources. Ph.D. thesis, Ecole nationale supérieure Mines-Télécom Atlantique (2017)
13. Lim, W.Y.B., Ng, J.S., Xiong, Z., Jin, J., Zhang, Y., Niyato, D., Leung, C., Miao, C.: Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems* **33**(3), 536–550 (2022). <https://doi.org/10.1109/TPDS.2021.3096076>

14. Liu, P., Chaudhry, S.R., Huang, T., Wang, X., Collier, M.: Multi-factorial energy aware resource management in edge networks. *IEEE Transactions on Green Communications and Networking* **3**(1), 45–56 (2019). <https://doi.org/10.1109/TGCN.2018.2874397>
15. Marozzo, F., Orsino, A., Talia, D., Trunfio, P.: Edge computing solutions for distributed machine learning. In: 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCOM/CyberSciTech). pp. 1–8. IEEE (2022)
16. Pantazoglou, M., Tzortzakis, G., Delis, A.: Decentralized and energy-efficient workload management in enterprise clouds. *IEEE Transactions on Cloud Computing* **4**(2), 196–209 (2016). <https://doi.org/10.1109/TCC.2015.2464817>
17. Rafique, H., Shah, M.A., Islam, S.U., Maqsood, T., Khan, S., Maple, C.: A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing. *IEEE Access* **7**, 115760–115773 (2019). <https://doi.org/10.1109/ACCESS.2019.2924958>
18. Rey-Jouanchicot, J., Del Castillo, J.Á.L., Zuckerman, S., Belmega, E.V.: Energy-efficient online resource provisioning for cloud-edge platforms via multi-armed bandits. In: 2022 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW). pp. 45–50. IEEE (2022)
19. da Silva, M.D.M., Gamatié, A., Sassatelli, G., Poss, M., Robert, M.: Optimization of data and energy migrations in mini data centers for carbon-neutral computing. *IEEE Transactions on Sustainable Computing* **8**(1), 68–81 (2023). <https://doi.org/10.1109/TSUSC.2022.3197090>
20. Tian, Z., Li, H., Maeda, R.K.V., Feng, J., Xu, J.: Decentralized collaborative power management through multi-device knowledge sharing. In: 2018 IEEE 36th International Conference on Computer Design (ICCD). pp. 409–412. IEEE (2018)
21. Wang, S., Xin, N., Luo, Z., Lin, T.: An efficient computation offloading strategy based on cloud-edge collaboration in vehicular edge computing. In: 2022 International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT). pp. 193–197 (2022). <https://doi.org/10.1109/CCPQT56151.2022.00041>
22. Xiong, Z., Kang, J., Niyato, D., Wang, P., Poor, H.V.: Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based admn for pricing. *IEEE Transactions on Services Computing* **13**(2), 356–367 (2019)
23. Zahoor, S., Javaid, N., Khan, A., Ruqia, B., Muhammad, F.J., Zahid, M.: A cloud-fog-based smart grid model for efficient resource utilization. In: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). pp. 1154–1160 (2018). <https://doi.org/10.1109/IWCMC.2018.8450506>
24. Zahoor, S., Javaid, S., Javaid, N., Ashraf, M., Ishmanov, F., Afzal, M.K.: Cloud-fog-based smart grid model for efficient resource management. *Sustainability* **10**(6), 2079 (2018)
25. Zhang, W., Zhang, Z., Zeadally, S., Chao, H.C., Leung, V.C.M.: Energy-efficient workload allocation and computation resource configuration in distributed cloud/edge computing systems with stochastic workloads. *IEEE Journal on Selected Areas in Communications* **38**(6), 1118–1132 (2020). <https://doi.org/10.1109/JSAC.2020.2986614>