applied sciences

MDPI

Article

Joint Task Offloading, Resource Allocation, and Load-Balancing Optimization in Multi-UAV-Aided MEC Systems

Ibrahim A. Elgendy [1,*,†], Souham Meshoul [2,*,†] and Mohamed Hammad [3]

[1] Department of Computer Science, Faculty of Computers and Information, Menoufia University, Shibin El Kom 32511, Egypt
[2] Department of Information Technology, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[3] Information Technology Department, Faculty of Computers and Information, Menoufia University, Shibin El Kom 32511, Egypt
* Correspondence: ibrahim.elgendy@ci.menofia.edu.eg (I.A.E.); sbmeshoul@pnu.edu.sa (S.M.)
† These authors contributed equally to this work.

Abstract: Due to their limited computation capabilities and battery life, Internet of Things (IoT) networks face significant challenges in executing delay-sensitive and computation-intensive mobile applications and services. Therefore, the Unmanned Aerial Vehicle (UAV) mobile edge computing (MEC) paradigm offers low latency communication, computation, and storage capabilities, which makes it an attractive way to mitigate these limitations by offloading them. Nevertheless, the majority of the offloading schemes let IoT devices send their intensive tasks to the connected edge server, which predictably limits the performance gain due to overload. Therefore, in this paper, besides integrating task offloading and load balancing, we study the resource allocation problem for multi-tier UAV-aided MEC systems. First, an efficient load-balancing algorithm is designed for optimizing the load among ground MEC servers through the handover process as well as hovering UAVs over the crowded areas which are still loaded due to the fixed location of the ground base stations server (GBSs). Moreover, we formulate the joint task offloading, load balancing, and resource allocation as an integer problem to minimize the system cost. Furthermore, an efficient task offloading algorithm based on deep reinforcement learning techniques is proposed to derive the offloading solution. Finally, the experimental results show that the proposed approach not only has a fast convergence performance but also has a significantly lower system cost when compared to the benchmark approaches.

Keywords: task offloading; resource allocation; load balancing; mobile edge computing (MEC); unmanned aerial vehicle (UAV); reinforcement learning; optimization

check for updates

Citation: Elgendy, I.A.; Meshoul, S.; Hammad, M. Joint Task Offloading, Resource Allocation, and Load-Balancing Optimization in Multi-UAV-Aided MEC Systems. *Appl. Sci.* **2023**, *13*, 2625. https://doi.org/10.3390/app13042625

Academic Editors: Joon-Min Gil and Jisu Park

Received: 4 January 2023
Revised: 10 February 2023
Accepted: 15 February 2023
Published: 17 February 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Wireless sensors and Internet of Things (IoT) devices are now common and ubiquitous in today's world, and their application and utility have expanded significantly. Furthermore, with the introduction of stable, high-speed internet (e.g., in 5G and 6G), many new services and applications in the fields of virtual/augmented reality, facial recognition, video games and video streaming, e-health, vehicular networks, and natural language processing [1,2] have emerged. However, these services and applications require increased computing capacity and energy efficiency, which creates new challenges for these devices to address due to their limited computation and battery capacity [3,4]. As a result, task offloading is proposed as a prominent solution to these limitations, in which intensive and delay-tasks could be offloaded and then executed remotely at more powerful devices [5,6].

Mobile cloud computing (MCC) is regarded as a valuable solution for offloading the intensive tasks of IoT devices to cloud computing resources [7–9]. Due to the centralization of cloud computing resources, the implementation of this solution results in high latency. MCC is also confronted with the issue of security [10,11]. As a result, researchers tend to

distribute cloud computing resources to the nearby edge of IoT users, resulting in a new paradigm known as mobile edge computing (MEC), which efficiently addresses latency and security issues [12,13]. On the other hand, due to their constant use in a variety of fields, Unmanned Aerial Vehicle (UAV) systems have seen an increase in popularity and attracted considerable attention in recent years. Furthermore, UAVs' low deployment cost and high mobility make them popular and appealing devices for aided communication networks, acting as relay nodes [14], flying base stations [15], or terminal nodes [16].

Several models and approaches have been proposed recently for using computational offloading with UAV-assisted MEC systems [17]. From one perspective, some models focus on a single objective, while others cover multiple objectives [18]. From another perspective, some approaches address the task offloading for a single edge while others cover for a multiple edge server with and without the cloud [19]. Nevertheless, most of these solutions allow the mobile users to only offload their tasks to stations that are connected, implying an unbalanced load at stations [12,20]. Consequently, it may be difficult for some mobile users to complete their computation tasks within the acceptable latency threshold. Meanwhile, implementing an optimal strategy for multi-user in dynamic and complex systems such as multi-users MEC system is another challenging issue that should be carefully addressed. Motivated by these considerations, we propose an efficient technique for multi-user and multi-edges systems in this paper to balance base station loads and reduce system costs. A novel and efficient offloading algorithm is also designed to find a near-optimal solution. The main contributions reported in this paper are summarized as follows:

- An efficient load-balancing algorithm is introduced for optimizing the load among GBSs, in which the mobile users are redistributed to the most appropriate GBSs regarding their location, task size, and CPU cycles. In addition, UAVs are utilized as a potential MEC server to provide computation and communication resources by hovering in overcrowded areas where the GBSs server is still overloaded.
- Task offloading, load balancing, and resource allocation are jointly optimized for multi-tiers UAV-aided MEC systems via a formulation of an integer programming problem with the primary objective of minimizing system cost.
- A novel form of deep reinforcement learning is introduced in which the application task's requirements represent the system state and the offloading decision is used to define the action. The solution is then derived using an efficient distributed deep reinforcement-learning-based algorithm.
- Simulation findings demonstrate that the proposed model not only exhibits a fast and effective convergence performance but also significantly decreases the system costs compared with a benchmark approach.

The rest of this paper is structured as follows. In Section 2, the work pertaining to task offloading models will be highlighted, whereas the system model and the problem formulation are introduced in Section 3. Following that, in Section 4, we develop a novel and efficient task offloading algorithm based-deep reinforcement learning to derive the offloading solution. Afterward, experimental results are presented and discussed in Section 5. Finally, the conclusion and the future recommendations are presented in Section 6.

2. Related Work

In the last decade, the computing power of Unmanned Aerial Vehicles (UAVs) has improved by several orders of magnitude. UAVs have emerged as a promising technology for supporting a wide range of human activities, including target tracking, water quality inspection, disaster relief, and power patrol inspection to surveillance due to their inherent mobility and increasing computing power. As a consequence, there is significant interest in embedding UAVs into edge computing systems to provide value-added edge computing services. The recent proliferation of IoT devices, in conjunction with edge computing, have uncovered numerous opportunities for novel UAV-based applications. Due to their limited battery life and processing power, however, UAVs face difficulties when performing computationally intensive tasks. Multiple research proposals on UAV-assisted MEC

systems have been proposed. Learning and optimization have been widely employed in the development of such systems. In addition, single-UAV edge computing systems may not be adequate for serving remote users and accommodating diverse UAV application scenarios. Therefore, solutions based on multiple UAVs have been proposed to handle this issue.

The authors of [21] proposed iTOA, an intelligent task offloading approach for UA MEC networks, to help in determining which offloading tasks would alleviate the burden on the UAV computation platform and be most beneficial to it. The key features of their work are the use of the deep Monte Carlo tree search (MCTS) scheme and a splitting deep neural network (sDNN). The role of MCTS is to find the optimal offloading trajectory that maximizes some reward functions having to do with latency. More specifically, the problem has been formulated as an average service latency minimization task. In order to make room for the application of MCTS, the task offloading problem has been recast as a Markov decision process (MDP). The authors further integrated MCTS with a Long Short-Term memory (LSTM) network to better predict the quality of channels, whereas the second serves to hasten search convergence by providing prior probabilities to the MCTS being used. The simulation findings show that the proposed iTOA has 33% and 60% lower latency than the game theory and greedy search benchmark task offloading methods, respectively.

Similarly, the work presented in [22] uses MDP and reinforcement learning to address the MEC task offloading problem in a UAV patrol system. To that purpose, a two-stage Stackelberg game model has been developed to simulate the system, which includes UAVs and network edge nodes. The problem has also been described as an optimization problem that utilizes a multi-agent deep deterministic policy gradient (MADDPG) scheme to find its solution. Extensive simulations were conducted in order to show the effectiveness of the suggested approach in terms of delay, utility of the UAVs, and system performance when compared to a random approach, a non-dominated sorting genetic algorithm, and a QoS priority algorithm. In another scenario, UAVs have been utilized to offer IoT devices with MEC services, particularly in scenarios such as terrestrial signal blockage and shadowing that make it difficult for IoT devices on the ground to reach edge clouds.

In this regard, the authors of [23] addressed the requirement of MEC quality of service and UAV battery limited lifetime and provided a system that takes into account IoT job offloading and UAV placements. The topic is formulated as a non-convex problem that minimizes the weighted sum of service delay of IoT devices and energy consumption of UAV. To tackle the problem, the authors employed an algorithm based on successive convex approximation. The authors carried out a number of experiments to show that their collaborative UAV-EC framework outperforms baseline methods that rely only on UAVs or ECs for MEC in IoT.

The authors of [24] explore scenarios in which complex missions with interdependent tasks must be carried out using multiple UAVs. They proposed a multi-agent reinforcement learning approach for determining a close-optimum offloading policy of bandwidth and task allocation to minimize the computing missions' average response time while taking the dynamic nature of the environment and the UAVs' energy constraints into account. A MDP formulation has also been adopted, in which the policy of offloading is used as the action, and the reward is tied to response time performance. Three distinct topologies for task interdependencies in complex missions are investigated as well as the impact on the offloading policy. In their experimental work, the authors demonstrated a good convergence ability and a significant reduction in response time of complex missions.

Considering the scenario of large-scale sparely distributed user equipment, Zhao et al. [25] proposed a collaborative framework based on multi-agent deep learning to jointly determine the UAVs' trajectories, communication resource management of UAVs and computation task allocation in a way that minimizes the total of execution delays and energy consumptions in multi-UAVs MEC systems. Furthermore, to deal with the high dimensional action space and find the best offloading policy, a multi-agent twin delayed deep deterministic policy gradient algorithm is used. The framework's implementation

resulted in results that show a significant reduction in total system cost when compared to other approaches in both fixed and mobile user equipment scenarios.

Multiagent reinforcement learning was also employed in [26] to assist in decision making regarding task offloading from UAVs to edge clouds while minimizing the overall latency perceived by the user and UAV energy consumption. The distributed architecture presented in this paper made it possible to achieve the two aforementioned goals.

He et al. [27] emphasized the importance of fully utilizing the UAVs computation capabilities for effective remote edge computing in a multi-UAV MEC context. To that end, the authors proposed a multi-hop task offloading scheme with on-the-fly computation in which UAVs perform multi-hop task offloading collaboratively. Furthermore, UAVs use their local processors to perform some tasks. To jointly optimize deployment strategies and resource allocation, two distributed algorithms with linear complexity have been proposed. In the experimental study, both special cases and general cases are considered, and simulation results indicate that better performance in terms of the overall rate of a multi-UAV network has been improved through applying the proposed framework compared to the baseline scheme.

In [28], a two-layer optimization strategy is proposed to jointly optimize UAV task scheduling at the first layer via a dynamic programming-based bidding optimization method and bit allocation with UAVs trajectories while resolving the potential path conflict problem. According to simulation results, the proposed strategy outperforms greedy and random strategies in terms of total energy consumption by the user. The proposed system was able to eliminate UAV trajectory conflicts while satisfying safety constraints. Meanwhile, Yang et al. [29] have addressed the load-balancing problem for multi-UAV-aided mobile-edge computing. Specifically, UAVs are used as MEC nodes to provide the computation and storage capabilities for IoT devices. In addition, a new differential evolution (DE)-based technique is proposed to redeploy the UAVs to the best location according to the IoT distributions. Moreover, a deep learning-based approach is designed to solve the task-scheduling problem at each UAV and thereby improve task performance. However, the approach in [29] assumed that UAVs have large enough computation capabilities to address IoT devices, which is not true in the real world, especially, in a large-scale environment. Consequently, this impedes the capability of the system to effectively address real-time applications when the number of IoT devices increases. In contrast, in our work, UAVs are used as assistance to MEC nodes to address the load problem.

To reduce the ground nodes' energy consumption, the authors of [30] investigated collaborative multi-task computation and cache offloading for UAV-aided MEC networks, taking into account the time-sensitive tasks' Quality of Experience. To handle the multiple problem variants, a block coordinate descent-based solution is proposed. Three manageable subproblems were identified and solved iteratively, including trajectory optimization, UAV resource allocation (including computation capability and bandwidth), and offloading decisions that take into account task type and ratio. Multiple experiments were conducted in a simulated environment, and the authors reported that the results provided insight into the effectiveness of collaborative offloading as it was able to manage two jobs (computing and caching) while reducing the overall GN energy consumption and meeting the QoE requirements of various task types. Meanwhile, Zhou et al. [31] proposed a cooperative task offloading and resource allocation algorithm (CTROAA) that uses Lyapunov optimization to minimize energy consumption while taking system performance into account. The authors considered hybrid energy sources for multi-clouds with UAV assistance. Three critical subproblems pertaining to task offloading control, local computing control, and charging control and cloud computing have been considered. The first two problems were mathematically formulated as convex optimization problems, whereas the third problem was mathematically formulated as a combinatorial optimization problem and was solved using Simulated Annealing (SA). To demonstrate the efficacy of the algorithm, a mathematical analysis was conducted where authors demonstrated that CTORAA can achieve the arbitrarily defined profit–stability trade-off. In addition, simulation results were conducted

to demonstrate that the algorithm could adapt to varying task arrival rates, assure the stability of the queue, and outperform two benchmark approaches, "Fixed" and "Random" in finding the optimal solution to minimize energy consumption.

Elsewhere, the task-offloading problem for a multi-Internet of vehicles environment has been added in [32,33]. Specifically, He et al. [32] modeled task offloading, security, and resource allocation as a multi-objective problem for UAV-assisted vehicular ad hoc networks, where minimizing the task delay is the main goal. Afterward, a relax-and-rounding and Lagrangian-based algorithm is proposed to solve this problem in an effective manner. Whereas, in [33], an integrated problem of task offloading, RSUs' cooperation, and tasks' division is proposed for MEC-based vehicular networks, where minimizing the task delay and increasing the performance of the service are the main goal. Additionally, an efficient routing mechanism is introduced to boost service reliability and decrease the rate of session failures. Moreover, a model-based deep neural network is developed to solve this problem and derive the optimum solution.

Furthermore, recent task offloading and resource allocation models have been proposed for MEC networks [34,35]. Specifically, Mohamed et al. [34] proposed a multi-tiered edge-based resource allocation optimization framework for the heterogeneous execution of tasks. This framework can facilitate the different operations of offloading over diverse IoT environments. Moreover, an optimization strategy is proposed with the goal of reducing energy consumption while promoting processing computations, task execution time, and network bandwidth. Whereas in [35], a resource allocation has been investigated for multi-UAV-aided MEC networks, in which a mixed-integer programming problem is formulated with the goal of decreasing the overall system cost in terms of latency and energy. Moreover, an efficient algorithm based on deep reinforcement learning is proposed that joined UAV movement control, MU power control, and MU association to solve this problem and derive the solution.

Recently, the task offloading and resource allocation optimization for a mobile edge computing environment has been proposed in [36–38]. More specifically, Xu et al. [36] proposed a cooperative task-offloading approach for UAV-aided MEC systems, where the nearest mobile device of a UAV can be served as an assisted hob to transfer the tasks of far mobile devices to UAV for remote processing. In addition, a block coordinate descent-based algorithm is proposed to optimize the trajectory of UAVs and decrease the overall energy consumption of mobile devices and UAVs. Meanwhile, in a different contribution, a multi-task offloading and resource allocation approach was proposed for MEC systems in satellite IoT [37], in which the task allocation and scheduling are initially handled by multiple unmanned aerial vehicle-based air base stations, with edge computing provided by satellites. Moreover, the directed acyclic graph is utilized to drive the main dependencies between tasks, and then, an attention mechanism and proximal policy optimization collaborative-based algorithm is proposed to obtain the offloading strategy. Whereas, in [38], an energy and delay-optimized trajectory planning framework is proposed for a multi-UAV multi-IoT network. More specifically, Banerjee et al. designed a multi-UAV multi-IoT system environment in hexagonal cells, in which each cell involves a set of IoT devices and a single UAV. In addition, each cell has IoT devices grouped into clusters that UAVs hover over to collect and delegate tasks when needed. Moreover, the major objective of this study is the optimization of energy consumption and transition times between hovering points. Finally, to identify the Pareto-optimal front and choose the optimal solution, a multi-objective optimization technique is applied.

Table 1 provides a concise summary of the literature reviewed in terms of the main objective, solution, application and tier environments as well as highlights the main weakness.

It is observed from the above summary of related works that numerous efforts and approaches have been investigated for addressing the task-offloading issues for multi-user MEC systems with single and/or multiple edge servers either using the cloud or not. Nevertheless, most of these solutions allow mobile users to only transmit their tasks to the connected MEC server, which implies an unbalanced load on the edge servers.

Consequently, some mobile users may find it difficult to complete their computation tasks within the acceptable latency threshold. Meanwhile, implementing an optimal strategy for multi-usersa in dynamic and complex systems such as a multi-users MEC system is another challenging issue that should be carefully addressed. Motivated by these considerations, an efficient load-balancing model for multi-user and multi-edge systems is presented in this paper in order to balance the loads among base stations and reduce system costs. Furthermore, a novel task-offloading approach based on deep learning techniques is proposed to efficiently derive the offloading solution.

Table 1. A related work comparison.

Reference	Objective	Proposed Approach	Application	Tier Environment		Weakness
				Two	Multiple	
[21]	Minimize Average Service Latency	An intelligent task-offloading approach	UAV Edge-Cloud Network		✓	• Energy consumption of UAVs is addressed.
[22]	Maximize Offloading Revenue	A two-stage Stackelberg game model	UAV Edge-Cloud Network	✓		• Data privacy is not considered. • Load balancing problem among UAVs is not considered.
[23]	Minimize IoT Devices' Service Delay and UAVs' Energy Consumption	An innovative UAV-enabled MEC model	IoT Devices Network	✓		• Concentrate only single offloading request scenario. • Load balancing issue is not addressed.
[24]	Minimize Average Mission Response Time	A multi-agent reinforcement learning approach	IoT Devices Network	✓		• Load balancing issue is not addressed.
[25]	Minimize Execution Delays and Energy Consumption	A collaborative multi-agent deep learning-based framework	IoT Devices Network	✓		• Load-balancing issue is not addressed. • Mobility and privacy are not handled.
[26]	Minimize User's Latency and UAV Energy Consumption	A distributed multi-agent reinforcement learning-based technique	UAV Edge-Cloud Network		✓	• Mobility and privacy are not handled. • Balancing among UAVs is not considered.
[27]	Maximize Network Computing Rate	A multi-hop task offloading scheme	IoT Devices Network	✓		• Energy consumption for IoT devices is not handled. • Balancing among IoT devices is not considered.
[28]	Minimize User Energy Consumption	A two-layer task scheduling and dynamic service optimization strategy	IoT Devices Network	✓		• Execution time for IoT devices is not measured. • Mobility and privacy are not considered.
[29]	Optimize Average Slow-Down for Offloaded Tasks	A multi-UAV-enabled load-balancing scheme	IoT Devices Network	✓		• Insufficient for real-time applications with large scale. • Processing deadline for IoT tasks' is not considered.
[30]	Reduce Ground Nodes' Energy Consumption	A collaborative multi-task computation and cache offloading scheme	IoT Devices Network	✓		• Mobility is not addressed. • Interference resulting from the users' transmission is disregarded.
[31]	Minimize UAVs' Power Consumption under Stability Queue Constraint	A collaborative task offloading and resource allocation scheme	IoT Devices Network	✓		• Mobility and execution time are not addressed. • Balancing among IoT devices is disregarded.
[32]	Minimize Task Delay	An MEC-enabled UAV-assisted vehicular ad hoc network architecture	Vehicular Edge Computing Network		✓	• Energy consumption is not considered.
[33]	Minimize Task Delay and Increase Service Performance	An integrated model of task offloading, RSUs' cooperation, and tasks' division	Vehicular Edge Computing Network	✓		• Energy consumption is not considered. • Privacy and deadlines for tasks are not considered.
[34]	Minimize Energy Consumption and Promote Processing Computations, Network Bandwidth, and Task Execution Time	A multi-tiered edge-based resource allocation optimization framework	IoT Devices Network		✓	• Interference resulting from the users' transmission is disregarded. • Security and privacy for tasks are not considered.
[35]	Decrease the Energy Consumption and System Latency	A resource allocation strategy for multi-UAC-aided MEC networks	UAV-Aided MEC Network	✓		• Interference resulting from the users' transmission is disregarded. • The load-balancing issue is not addressed.

Table 1. *Cont.*

Reference	Objective	Proposed Approach	Application	Tier Environment		Weakness
				Two	Multiple	
[36]	Minimize Overall Energy Consumption of Mobile Devices and UAVs	A cooperative task offloading and resource allocation approach for UAV-aided MEC systems	UAV-Aided MEC Network	✓		• Execution time for mobile devices is disregarded. • Interference resulting from the users' transmission is disregarded.
[37]	Minimize the Total Cost of All Tasks	A multi-task offloading, and resource allocation approach	MEC Systems in Satellite IoT		✓	• Load-balancing issue is not addressed. • Privacy for tasks is not considered.
[38]	Optimize Energy Consumption and Transition Times between Hovering Points	An energy and delay-optimized trajectory planning framework	UAV-Aided MEC Network	✓		• Load-balancing issue is not addressed.

3. System Model and Problem Formulation

This section starts by introducing the multi-UAV-aided MEC system model. Following this, optimization problems pertaining to task offloading, resource allocation, and load-balancing models are formulated with the aim of minimizing the total system cost.

3.1. System Model

In this study, we consider a multi-UAV-aided MEC system as shown in Figure 1, which consists of three main layers. In the first layer, a $\mathcal{D} = \{1, \ldots, K\}$ set of device users is distributed in which each device has an intensive application required to be executed. In the second layer, there is a $\mathcal{G} = \{1, \ldots, M\}$ and $\mathcal{U} = \{1, \ldots, N\}$ set of ground base stations (GBSs) and UAVs, respectively, that offer the storage and computation capabilities for device users due to their limitations in computation and battery. Moreover, UAVs can be utilized as a potential MEC server to provide communication and computation resources through hovering over the crowded areas which are still loaded due to the fixed location of GBSs. Furthermore, UAVs and GBSs are controlled and managed through a backbone router, in which the controller technology of software defined network (SDN) is efficiently utilized. Finally, in the last layer, a single cloud is configured and attached to the backbone router in the second layer through the core of the network.

Consequently, each device user can process their intensive computation application locally using their resources or offloading and then processing at one of the available GBSs or UAVs (in the case of GBSs being overcrowded) or at the cloud server. Thereupon, $\mathcal{S} = \{0, 1, \ldots, M, M+1, M+2, \ldots, J, J+1\}$ is used to denote the set of the available servers that can be used to process the intensive application of mobile user, where 0 indicates the local resources of mobile users, from 1 to M indicates one of the available GBSs, from $M+1$ to J (i.e., $M+N$) indicates one of the available UAVs, and $J+1$ indicates the cloud server.

Moreover, let $y_{i,j} \in \{0, 1\}$ be a binary offloading decision for the intensive application of mobile i that will be allocated to be processed at server j, where $i \in K, j \in S$. More specifically, ($y_{i,0} = 1$) means the mobile i decides to process its application locally, and ($y_{i,J+1} = 1$) means the mobile i decides to offload and process its application remotely at the cloud server; otherwise, the mobile i decides to offload and process its application remotely at GBSs or UAVs. Overall, the application of each mobile user must be processed by only one server (including itself, i.e., server 0). Therefore, $\sum_{j=0}^{J+1} y_{i,j} = 1$, where the offloading servers value depends on:

$$\begin{cases} y_{i,0} = 1 & Local\ Execution \\ \sum_{j=1}^{J} y_{i,j} = 1 & GBSs\ or\ UAVs\ Execution \\ y_{i,J+1} = 1 & Cloud\ Execution \end{cases} \quad (1)$$

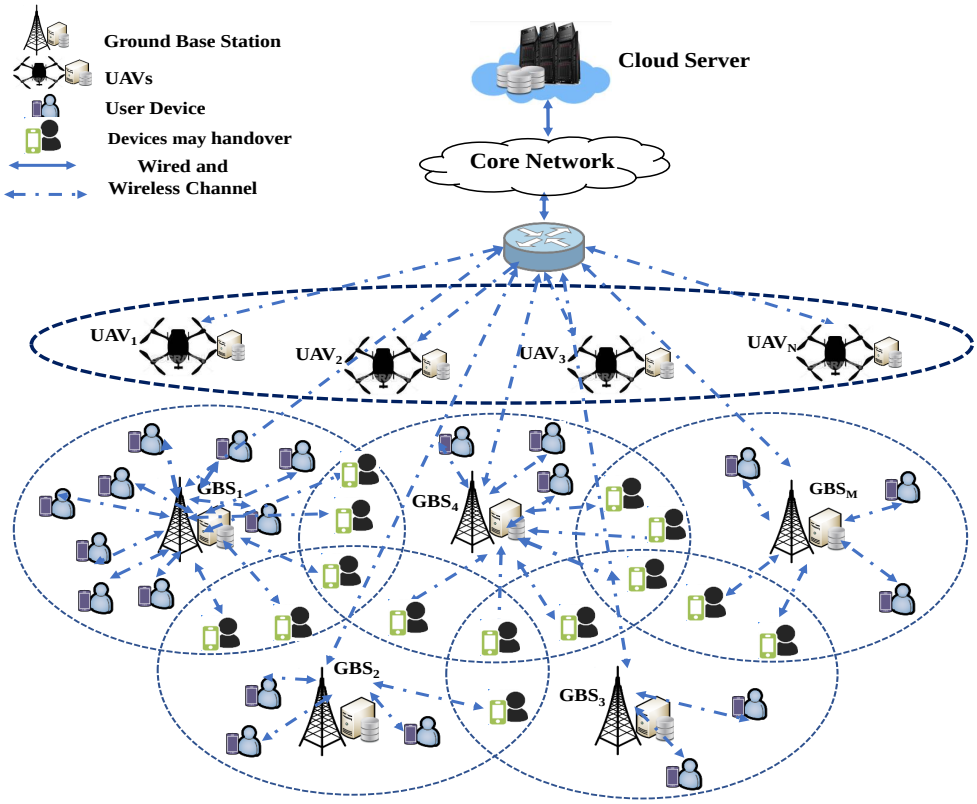


Figure 1. System Model.

Based on previous studies in MEC [39,40], we adopt a quasi-static model in our simulations where the devices' number remains unchanged over the offloading period, whereas it may be changed over the different periods. A further discussion of the load balancing, communication and computation setup and their requirements for mobile-edge cloud computing models are presented below, taking into account the key roles they play.

3.1.1. Load Balancing

In this subsection, we investigate the design of the load-balancing process among GBSs using UAVs. First, at time t, the distribution of mobile devices across the GBSs is unbalanced, with some GBSs being overloaded while others are underloaded, as depicted in Figure 1. Consequently, the network congestion degrades the service quality and application's latency for these devices. Therefore, the main contribution of load balancing there falls into how to balance the load among GBSs. This can be achieved in two main phases. These can be accomplished as follows. In the initial phase, we redistribute the reallocated IoT devices (i.e., devices that exist in the intersection area of GBSs) to be connected to the best GBSs (i.e., least loaded) by compelling them to hand over. Then, in the second phase, we locate the GBSs that are still overcrowded based on a predetermined threshold value θ and then assign one or more UAVs hovering above to alleviate their loads through providing the computing and storage capabilities for their mobile users. The detailed process steps for balancing the load among GBSs are as follows.

Initially, a summary of the information about the mobile devices associated with each GBS is sent to the manager through the connected GBSs. This information includes the total number of mobile devices, the available rate of data for each user, the CPU cycles and data size required for each task associated with the mobile device and all the mobile devices that exist around intersection area of GBSs (i.e., black mobile users in Figure 1) and may be reallocated to another nearby GBS. The central manager then iterates over all mobile devices in the intersection area and, based on the collected information, forces them to hand over to the most appropriate available GBS. Then, user's computation capabilities and

data rate are updated once the appropriate GBS is chosen for each mobile device. After all mobile devices have been assigned to the optimal GBS, the steps are repeated. Subsequently, upon receiving the updated information, the central manager determines the new number of mobile users at each GBS and finds the GBSs that are still overloaded (i.e., the number of users greater than given threshold θ). It then hovers one of the available UAVs above these GBSs so they can be able to provide new computation and storage capabilities. By doing so, the overhead consumption for mobile devices will be minimized. Algorithm 1 outlines the comprehensive process for balancing the load among the GBSs.

A snapshot example of IoT devices' distribution across GBSs is shown in Figure 2 to illustrate the algorithm execution. We can observe from this figure that seventeen IoT devices are distributed across three GBSs, where 12 devices are connected with GBS_1 and 2, and three devices are connected with GBS_2 and GBS_3. Additionally, D_{10}, D_{11}, D_{12}, D_{15}, and D_{16} are all co-located across GBSs' boundaries, making it easy to allocate devices as needed. Furthermore, each GBS is capable of providing 20 GHz computation capabilities and a bandwidth of 20 MHz that can be shared with connected devices. Finally, our goal is to redistribute workloads among GBSs as well as provide UAV-enabled edge computing services in the still-overloaded area so that quality is improved and energy is reduced. Furthermore, redistributing the workloads among GBSs is possible and can be achieved through two main phases: handing over devices to the best-performing GBS and providing UAV-enabled edge computing services in the still-overloaded GBSs. Based on the given parameters' values, these phases can be technically achieved as follows.

Algorithm 1 GBSs Balancing Load

Initialization: Each mobile device i is located and associated with a given GBSs j.
/*Phase one – Redistribute mobile devices across GBSs*/
for all GBSs j and at given time slot t **do**
  $\zeta \leftarrow$ Numbers of mobile devices.
  $\varphi \leftarrow$ Mobile devices' requirements in terms of CPU cycles and task size.
  $\eta \leftarrow$ Calculate the computation capabilities and data rate assigned for each device at GBSs.
  $\vartheta \leftarrow$ Determine the mobile devices that can be reallocated to another GSBs.
  Determine the optimum GBSs for each mobile user and then force it to hand over with respect to the values ofζ, η, ϑ.
end for
/*Phase Two – Provide the Overloaded GBSs with UAVs*/
for all GBSs j and at given time slot t **do**
  $\zeta \leftarrow$ Find the updated mobile devices' numbers associated with each GBS.
  if $\zeta > \theta$ **then**
    Provide one UAV with computation and storage capabilities hovering above this GBS.
  end if
end for

As shown in Table 2, the central control manager first collects summary information about the current environment, such as the number of GBSs and their connected devices, the number of devices that can be reallocated, the data rate available, and the task requirements for each device, such as data size and CPU cycles. Then, it iterates over the reallocated devices and determines the optimal GBS based on the estimated execution time. For instance, D_{10} will be handed over to GBS_3 where the estimated time (i.e., upload and computation) is 23.2 s at GBS_1 and 6.8 s at GBS_3. In addition, D_{11} and D_{12} will be handed over to GBS_2 which reduces the time, whereas D_{15} and D_{16} remain connected to GBS_3. In our example, after the IoT devices are allocated to the best GBSs, some GBSs may still be overloaded (e.g., GBS_1 in our example). Therefore, we can proceed to the second phase, UAV deployment, which can be performed as follows. Based on the updated number of connected devices for each GBS, and according to the given threshold $/theta$

(e.g., $/theta = 6$), UAVs are hovering above the area of GBSs that have devices more than or equal to θ (i.e., GBS_1) and providing the computation capabilities to IoT devices.

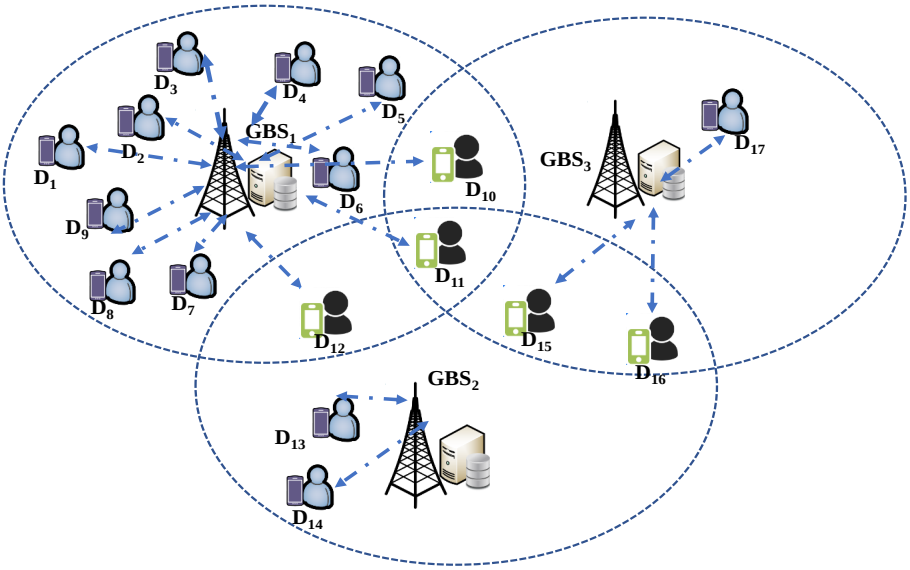


Figure 2. Snapshot of IoT devices distribution.

Table 2. Proposed algorithm illustration.

IoT Devices	Data Size (MB)	CPU Cycles (Gigacycles)	Estimated Execution Time (s)		Best GBS
			Upload Time	Computation Time	
D_{10}	15	15	$GBS_1 \rightarrow 14.4$ $GBS_3 \rightarrow 4.8$	$GBS_1 \rightarrow 8.8$ $GBS_3 \rightarrow 2.0$	GBS_3
D_{11}	20	10	$GBS_1 \rightarrow 17.5$ $GBS_2 \rightarrow 4.8$ $GBS_3 \rightarrow 8.0$	$GBS_1 \rightarrow 5.6$ $GBS_2 \rightarrow 1.5$ $GBS_3 \rightarrow 2.5$	GBS_2
D_{12}	15	20	$GBS_1 \rightarrow 12.0$ $GBS_2 \rightarrow 4.8$	$GBS_1 \rightarrow 10.0$ $GBS_2 \rightarrow 4.0$	GBS_2
D_{15}	10	15	$GBS_2 \rightarrow 4.0$ $GBS_3 \rightarrow 3.2$	$GBS_2 \rightarrow 3.75$ $GBS_3 \rightarrow 3.0$	GBS_3
D_{16}	15	25	$GBS_2 \rightarrow 4.0$ $GBS_3 \rightarrow 3.2$	$GBS_2 \rightarrow 3.75$ $GBS_3 \rightarrow 3.0$	GBS_3

3.1.2. Communication Model

In this subsection, we investigate the transmission time and energy consumption associated with communications between the mobile device and servers. Moreover, each intensive application is denoted by a tuple of $(\alpha_i, \mu_i, \beta_i, \Gamma_i)$ for $i \in K$, where α_i and μ_i, respectively, indicate the input and output data size of the application, β_i indicates the number of CPU cycles needed to accomplish the application, and Γ_i indicates the deadline associated with each application.

Consequently, in this paper, guided by the work in [41], the total energy and time consumption for returning the output data size is neglected because of its small size compared with the input data size. Moreover, to mitigate the uplink interference of multi-users transmission in the same cell, an Orthogonal Frequency Division Multiple Access approach is selected [42].

Following that, regarding the Shannon law [43], if the mobile user i decides to offload and then process its application remotely at GBSs or UAVs J, then the upload data rate among the mobile device and the edge can be identified as follows:

$$R_{i,j} = B_j log_2(1 + \frac{p_i g_0^2}{\omega_0 B_j}), \quad j \in 1...J \tag{2}$$

where B_J and p_i, respectively, indicate the bandwidth and transmission power of mobile i and g_0 and ω_0, respectively, indicate the associated gain and noise power.

If the mobile user i decides to offload and then process its application remotely at the cloud server $J+1$, then one of the available GBSs or UAVs will be chosen as a relay node to transfer the application's data and requirements. In this study, the GBSs or UAVs with the greatest uplink data rate are chosen as a relay node, and its equation can be identified as follows:

$$R_{i,J+1} = \max_{j \in \{1...J\}} R_{i,j} \tag{3}$$

Furthermore, if the mobile user i decides to process its application locally, for completeness, we assume that the uplink data rate is $R_{i,0} = \infty$.

Finally, the overhead consumption for the communication can be computed as follows:

$$T_i^{Comm} = T_i^U + \zeta y_{i,J+1} \tag{4}$$

$$E_i^{Comm} = p_i T_i^U \tag{5}$$

where ζ denotes the prorogation delay between the cloud and edge server, and T_i^U denotes the upload time that can be expressed as:

$$T_i^U = [\sum_{j=0}^{J+1} \frac{\alpha_i}{R_{i,j}} y_{i,j}] \tag{6}$$

3.1.3. Computation Model

In this subsection, we investigate the transmission time and energy consumption associated with processing each task at the available servers. Moreover, the available capabilities in CPU cycles for each server j are denoted by f_j, where $j \in \{0...J+1\}$.

Note that the computation capabilities at the cloud are more powerful than GBSs and UAVs and the capabilities of GBSs and UAVs are more powerful than those of mobile devices. In addition, in this article, we assume that the computation capabilities of GBSs and UAVs are equally shared between all the mobile devices that transmit their tasks to the same server. Moreover, the computation capabilities assigned for each device at server j can be expressed as follows:

$$f_i = \sum_{j=0}^{J+1} \frac{f_j y_{i,j}}{\sum_{i=1}^{K} y_{i,j}} \tag{7}$$

As a consequence, the time and energy spent on executing the task of each mobile i can be computed as:

$$T_i^{Comp} = \frac{\beta_i}{f_i} \tag{8}$$

$$E_i^{Comp} = \xi_i T_i^{Comp} \tag{9}$$

where ξ is a constant coefficient that denotes the consumption of energy where the mobile device is being idle.

Finally, the overhead for processing the task of each device user i, in which load balancing, communication, and computation models are considered, can be expressed as:

$$\Omega_i = w_i^t (T_i^{Comm} + T_i^{Comp}) + w_i^e (E_i^{Comm} + E_i^{Comp}) \tag{10}$$

where w_i^t and $w_i^e \in [0,1]$ are scalar weights denoting the time and energy consumption, respectively, which depends on the nature of the application. For instance, if $(w_i^t = 0)$ and $(w_i^e = 1)$, the mobile user targets an application with an energy-sensitive, or maybe the battery of the device is in a low state. Whereas, if $(w_i^t = 1)$ and $(w_i^e = 0)$, the mobile user targets an application that is time-sensitive. Additionally, for different objectives, w_i^t and w_i^e are set at different values. Moreover, through the application settings, these weights can be adjusted at any time.

3.1.4. Problem Formulation

The formulation of the problem of a multi-user, multi-UAV-aided MEC system is presented in this subsection. Additionally, load balancing, task offloading, and resource allocation are all taken into account for each mobile user. Moreover, both the energy and time overhead are jointly optimized in the objective. The problem can be formulated as follows:

$$\begin{aligned} \min_{a} \quad & \Big[\sum_{i=1}^{K} \Omega_i\Big] \\ \text{s.t} \quad & \Big[T_i^{comm} + T_i^{Comp}\Big] \leq \Gamma_i, && C1 \\ & \sum_{a=0}^{J+1} y_{i,j} = 1, && C2 \\ & y_{i,j} \in \{0,1\}, && C3 \end{aligned} \tag{11}$$

where decreasing the total system cost is the main objective. In addition, the first constraint handles the delay requirements for each user, whereas constraint C2 ensures that each task will only be executed once and constraint C3 assures the binarization of the task-offloading variable.

The problem's solution is derived through determining the best values for the offloading. Nevertheless, it is not possible to solve this problem using a convex feasible set since y is a binary variable. In addition, the problem is NP-hard, since the objective is not convex [44]. Moreover, solving it in polynomial time is difficult, particularly with a large number of device users, since the problem size grows exponentially with the increase in devices. As result, reinforcement learning can be used as an alternative to conventional optimization methods to solve these problems and obtain close-optimal solutions.

4. Deep Reinforcement Learning-Based Approach for Solving the Problem

Throughout this section, we demonstrate how deep reinforcement learning can be used for solving our optimization problem effectively, thereby reducing the time complexity and effort involved in solving it. First, we introduce the reinforcement learning definition and then highlight its key elements. After this, a distributed deep learning scheme is introduced to achieve a close optimum solution.

4.1. Reinforcement Learning Introduction

It is important to note that one of the most cutting-edge fields of machine learning, reinforcement learning (RL), is capable of coping with an unpredictable and dynamic environment and also taking a variety of actions in order to maximize the accumulated reward. More specifically, RL is comprised of five main elements, namely environment, agent, action, reward, and state space. Firstly, for a given environment and a specific time t, the state s is observed by the agent, and then based on the policy $\pi = P(a_t|s_t)$, an action is selected to move the agent from the state s_t to the next state $s+1$. The agent then applies the reward function $R(s,a)$ to earn a reward r. Finally, the agent repeats this procedure until it reaches the final state and maximizes the total reward regarding $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in [0,1]$ is the factor for discount.

4.2. Reinforcement Learning Key Elements

For the system model to be converted into reinforcement learning equivalents, the state, actions, and reward function must be defined, which represent the key elements of RL. In our environment, where multiple users are performing an intensive-application on an MEC system, the key elements of RL are specified as follows:

- State: In our study, the computational requirements for the intensive application can be utilized to define the state space S as follows $s_t = \{(\alpha_i, \beta_i, \Gamma_i)_t | i \in K\}$
- Action: The offloading decision can be utilized to specify the action space A, in which selecting an action $a_t = \{(y_i)_t | i \in K\}$ based on the s_t can follow the policy $\pi(a_t|s_t)$.
- Reward: The reward value is given by the objective function according to (Equation (11)) as part of our problem formulation. Therefore, the objective function value at time t can be calculated using policy $\pi(a_t|s_t)$ depending on the state s_t and after selecting an action a_t. Afterward, the same procedure is continuously repeated with the time index increasing as $t = 1, 2, \ldots, T$. As a result, based on the findings of this study, the total reward r_t is minimized using a policy π that can be defined as $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} r_t$, where r_t denotes the ω_i in Equation (11).

4.3. Distributed Deep Reinforcement Learning-Based Algorithm

It is vital to highlight that distributed deep q-learning is an extended version of the deep Q-learning algorithm that incorporates a series set of deep neural networks (DNNs) capable of parallel processing and thereby deriving the most appropriate solution efficiently [45]. This section presents a distributed deep reinforcement learning-based strategy that is presented to roughly decrease the total reward values shown in Equation (11).

The architecture of our proposed distributed deep reinforcement learning-based algorithm is shown in Figure 3, in which a B number of DNNs are used with a shared and fixed-size replay memory M. In addition, the application task's requirements are entered as input (i.e., system state) and the most appropriate offloading decision is obtained as output (i.e., offloading decision). More specifically, the system is provided by the state s_t and then each DNN can generate an offloading action y_t^b regarding $f_{w_t^b} : s_t \rightarrow y_t^b$, where $b \in B$ denotes as the index of DNN and $f_{w_t^b}$ represents the b^{th} DNN with the weight value w_t^b. Moreover, from all the generated actions, the action with the least reward value can be selected as output regarding $y_t^* = arg\min_{b\in B} Q(s_t, y_t^b)$ and then stored as transition experience (s_t, y_t^*) in replay memory M. Subsequently, according to the stored transitions, the DNNs can be trained and then updated via choosing a random sample of data from memory M.

In Algorithm 2, the procedure to derive the near-optimum task offloading solution is outlined. This procedure is as follows. First, we have B number of DNNs which are initialized with different random values of weights w_t^b, and the replay memory is assigned with an empty finite-size S. Then, at time t, the application task's requirements (i.e., α, β and Γ) is given as the input of state s_t. Afterward, each DNN obtains the system state s_t and generates a set of B actions according to $f_{w_t^b} : s_t \rightarrow y_t^b$, in which $b \in B$ denotes the index of DNN and $f_{w_t^b}$ represents the b^{th} DNN with the weight value w_t^b. Subsequently, based on the reward function $arg\min_{b\in B} Q(s_t, y_t^b)$, the action with the least reward y^* is chosen and then stored with the state s_t as transition value (s_t, y^*) in the memory. Lastly, the DNNs are then trained and updated by selecting random transitions from memory.

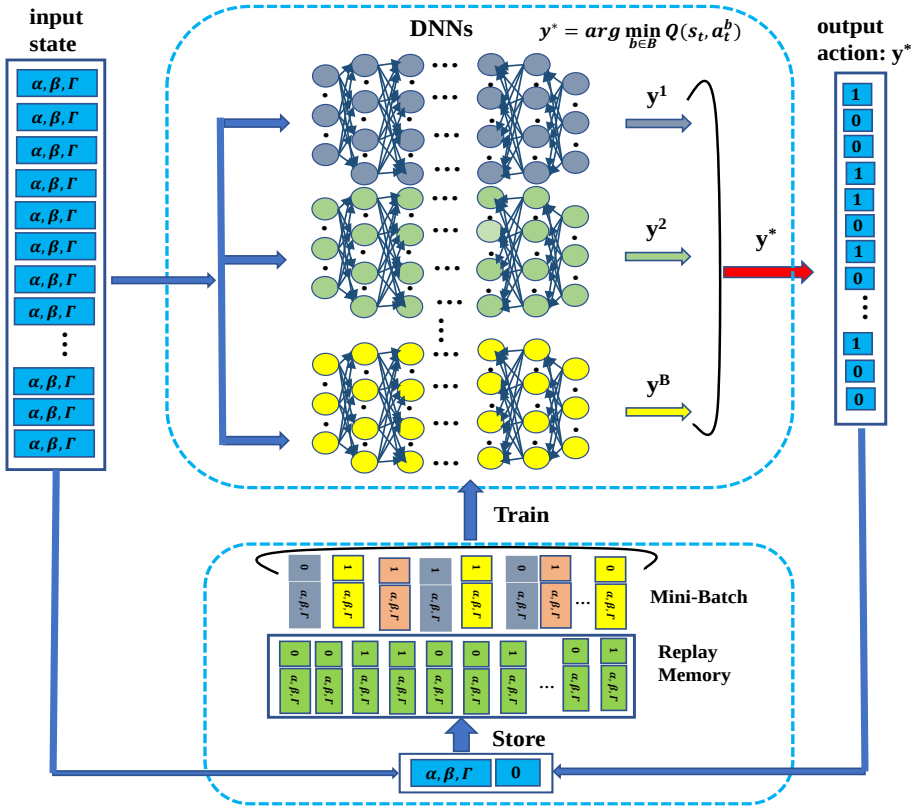


Figure 3. Architecture of the proposed distributed deep reinforcement learning-based algorithm.

Algorithm 2 Distributed Deep Reinforcement Learning-Based Algorithm

Initializing the DNNs with different random weights w_t^b.
Assigning the memory M with size S.
for $t = 1, 2, \ldots, G$ **do**
  Each DNN uses the same input state s_t.
  Generating a set of actions from the DNNs $\{a_t^b\} = f_{w_t^b}(s_t)$.
  Selecting the action with least value regarding $a_t^* = arg\min_{b \in B} Q(s_t, a_t^b)$.
  Storing the values of transition (s_t, a_t^*) in the memory M
  Selecting a random sample of transitions from the replay memory.
  Training and Updating DNNs.
end for

5. Performance Evaluation and Discussion

5.1. Experiment Setup

We conducted our simulation on a desktop computer with 16 GB of RAM and using an Intel® Core(TM) i7-4770 processor clocking at 3.4 GHz. Regarding the environment, there are 100 mobile devices distributed across five GBSs and three UAVs, and each mobile need to process an intensive application. In addition, there are 0.6×10^9 cycles per second on each mobile device, 10×10^9 cycles per second on edge servers, and 1×10^{12} cycle per second on the cloud server. The transmission power for the mobile device is set to 0.2 Watts, and 10 MHz of bandwidth is available between the mobile device and edge servers. Each mobile device has randomly assigned $(10, 30)$ MB as the input data size, which follows the random distribution and has also assigned with 1900 cycles per byte for the computation requirements. Moreover, a further analysis was made of the time

and energy for each mobile, which was determined to be 4.75×10^{-7} seconds per bit and 3.25×10^{-7} Joules per bit, respectively [46]. It is estimated that there is a propagation delay of 15 ms between the edge servers and cloud. We have set the weight for both the execution time as well as the energy consumption as $w_i^t = 0.5$, and $w_i^e = 0.5$, meaning that the each device is considering both metrics. Furthermore, regarding the deep-learning algorithm's parameters, four layers are considered in each DNN, in which two of them are hidden layers with 120 and 80 neurons and the episode size, mini-batch size, learning rate and memory size are set as 20,000, 32, 0.01, and 1024, respectively. Following these specifications, we ran 50 simulations round and calculated average values.

5.2. Experiment Results and Discussion

5.2.1. Convergence Performance of System

This subsection illustrates the convergence performance for our work, where the appropriate value for each parameter has been selected for use in the remaining simulation based on applying different values for each parameter.

First, the parameter of learning rate is optimized in Figure 4, in which different values are used with regard to the ratio of reward. It is seen from the plot that the performance convergence is observed to be faster with a value of 0.01 as well as the speed is increasing with the increase of learning rate value. Nevertheless, the convergence of performance speed for the large value (i.e., 0.1) drops, whereas a local optimum solution occurs. Therefore, we have decided that 0.01 would be the most suitable value for learning, since it is a strong indicator of how well the learner will be able to adapt their behavior in a given situation.

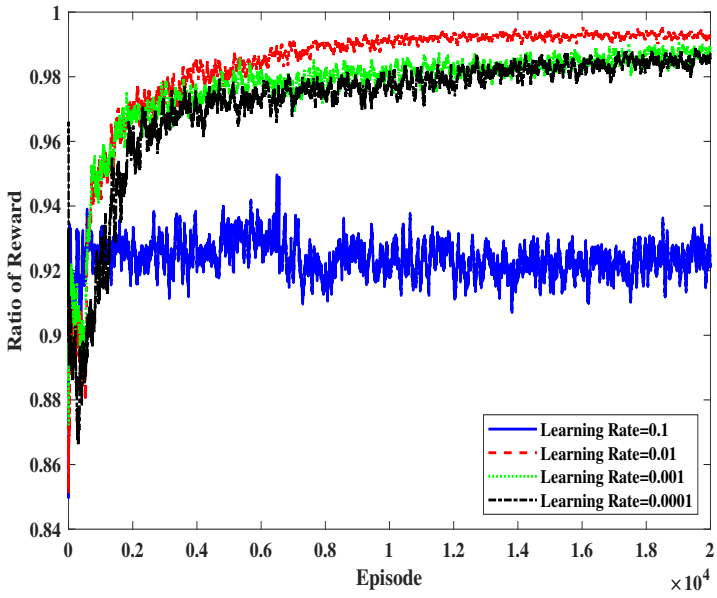


Figure 4. Convergence of performance over different learning rate values.

Secondly, the DNNs' parameter is optimized in Figure 5, in which different numbers are used with regard to the ratio of reward. It is clear that our suggested model can reach convergence more quickly as the number of DNNs increases. Moreover, with only three DNNs, the ratio of reward can reach 0.96 after 2000 steps of learning. Nevertheless, we observed that our proposed model could not converge well when only a few DNNs are used (i.e., DNNs = 1).

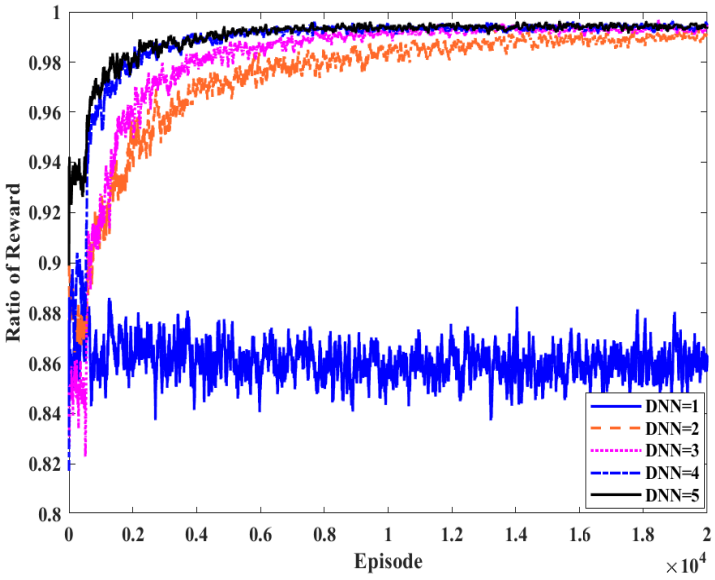


Figure 5. Convergence of performance over different number of DNNs.

Lastly, the parameter of batch size is optimized in Figure 6, in which different values are used with regard to the ratio of reward, where this parameter indicates the trained samples at each interval. The figure clearly demonstrates that the convergence with 32 is more rapid than with the other values. This is because the gradient descent direction tends to increase with decreasing mini-batch size values, leading to a more rapid updating of neural network weights. As a consequence, the batch size is determined to be 32, which appears to be optimal.

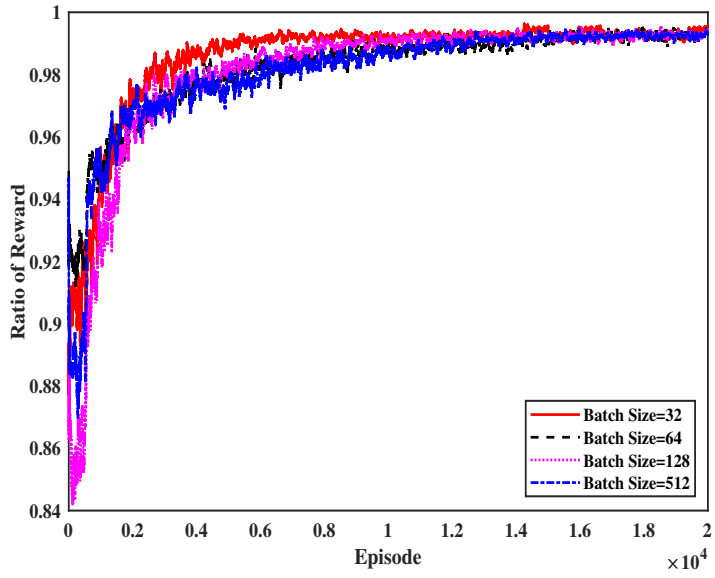


Figure 6. Convergence of performance over different number of batch sizes.

5.2.2. System Performance

As a means of demonstrating and validating the model, simulations were conducted under four different scenarios, which are as follows:

- Local Policy (LP): In this policy, there is no offloading. The application's tasks are carried out locally on the device resources.

- Full Offloading Policy (FOP): This policy involves offloading the application's tasks to GBSs for remote processing.
- Proposed Model Policy: In this policy, according to our proposed model, the application's tasks will be processed according to the offloading decision, which will minimize the total overhead of the system in the end.
- Task Offloading Policy (TOP [47]): This policy is set up to handle the application's tasks for mobile users based on the model proposed in [47] whereby each mobile user should send their application's tasks to the connected GBSs in the event that it does not take the selection of GBSs into consideration.

First, Figure 7 depicts a comparison of total cost for various numbers of mobile users under the four different policies. According to the graph, the proposed model has the lowest system cost when compared to the other policies. With a small number of users, the total cost of TOP and FOP policies approaches that of the proposed model. However, there is an increasing gap among them as the number of users increases. Furthermore, when the number of users exceeds 60, the cost of the FOP policy exceeds the cost of the LP policy. The reason for this could be that the GBSs lack the computation capabilities to handle more users connected to them at the same time due to shared channels, which is one of the reasons for redistributing users among GBSs and using UAVs as assisting nodes, which significantly impacts the system's performance.

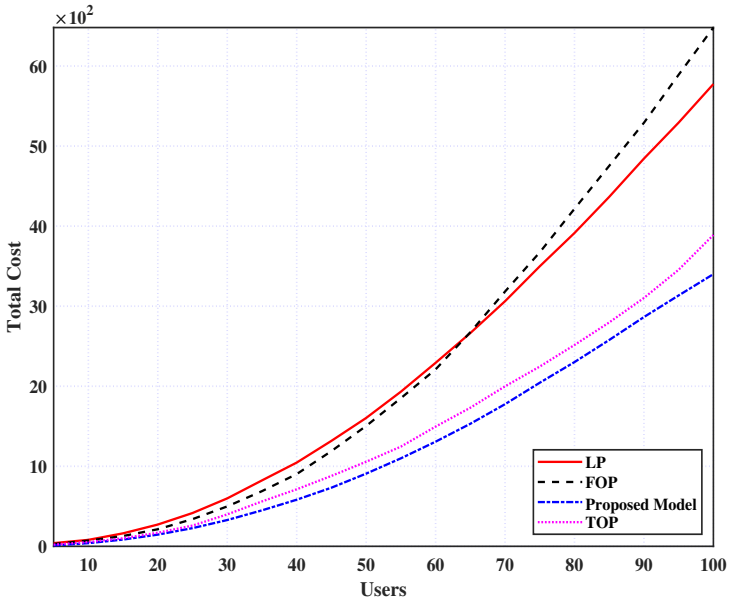


Figure 7. A comparison of total cost for different numbers of mobile users (M = 5, N = 3).

Second, a comparison of total cost for different numbers of GBSs under the four different policies is illustrated in Figure 8. In light of this figure, it can be observed that LP is unaffected by the GBSs' number, whereas the other policies' costs are steadily decreasing. The LP policy does not utilize GBSs resources, whereas mobile users are allocated with more resources, resulting in a shorter processing time and thereby system cost. Furthermore, selecting the right GBSs and UAVs to perform transmission and processing tasks also has a significant impact on the system performance.

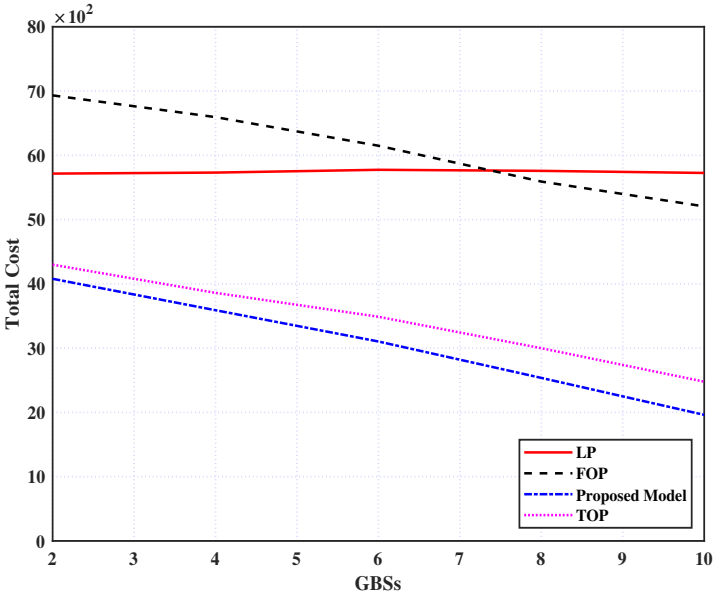


Figure 8. A comparison of total cost for different numbers of GBSs (K = 5, N = 3).

Furthermore, Figure 9 illustrates the comparison of the successful task processing ratio under different numbers of users, where this ratio denotes the proportion of successfully completed tasks relative to the total number of tasks. It is observed from the figure that the ratio is 100% for a small number of users (i.e., less than 20) for the three policies. However, it steadily degrades as the number of users increases and reaches 85% and 71% for TOP and FOP policies, whereas it slightly decreases for our proposed model and reaches 98% for 100 users. This variation can be explained by the fact that for TOP and FOP policies, as the number of users increases, the available resources at GBSs become competitive among users, whereas our model can balance the load among servers and UAVs can be used to efficiently utilize the computation resources of edge servers.

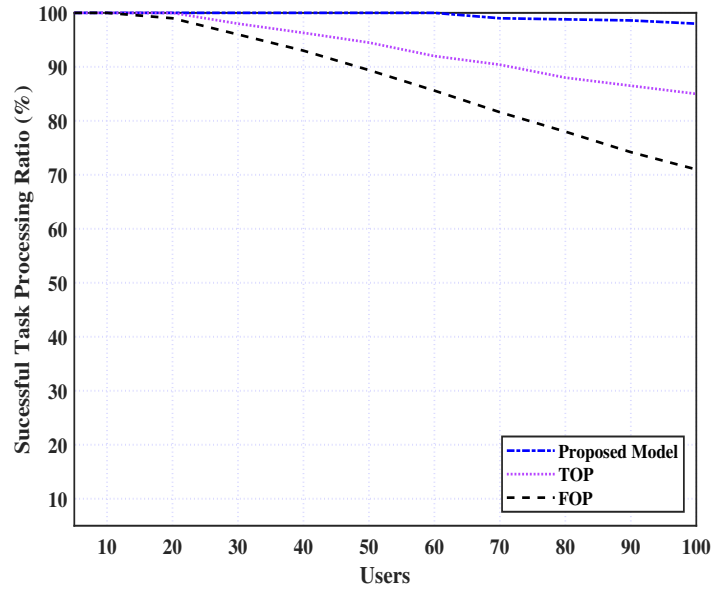


Figure 9. A comparison of successful task processing ratio under different numbers of users (K = 5, N = 3).

Finally, Figure 10 shows how the total cost of the four strategies was calculated for five different types of applications (shown in Table 3). The figure shows that the total cost

of the applications in categories A, B, and C for the FOP policy is higher than those in the other scenarios, whereas those in the LP policy for the other applications (D and E) are higher than those in the other policies. The reason for this is that the communication requirements for applications A, B, and C exceed the computation requirements. As a result, for these communication-intensive applications, LP policy is a better choice. D and E, on the other hand, are computationally intensive, so offloading policies are the best option. Furthermore, balancing the load among GBSs and employing UAVs as assistant nodes can improve the proposed model's performance compared with TOP policy.

Table 3. Applications complexity [48].

Application	Label	CPU Cycle/Byte
Health Monitoring	A	500
Automatic Number Plate Reading	B	960
x264 CBR encode	C	1900
Traffic Management	D	5900
Augmented Reality	E	12,000

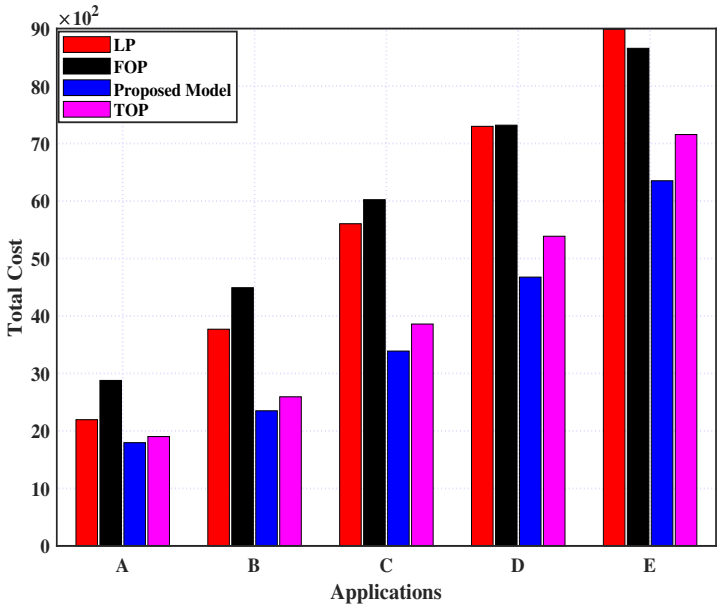


Figure 10. A comparison of total cost for different numbers of GBSs (K = 5, N = 3, K = 100).

6. Conclusions

For multi-user, multi-tier UAV-aided MEC systems, an integrated model of load balancing, resource allocation, and task offloading is proposed. In this model, an effective load-balancing model is designed to optimize the load among ground MEC servers by handing off users in the intersection area between GBSs to the most suitable one. In addition, UAVs are utilized as potential MEC servers to provide communication and computation resources by hovering over crowded areas where the ground-based MEC server is still overloaded. In addition, task offloading, load balancing, and resource allocation are jointly optimized via a formulation of an integer problem with the primary objective of minimizing system cost. This formulation is of the NP-hard variety, which is challenging to solve in polynomial time. For this problem, a novel form of deep reinforcement learning is presented in which the application task's requirements represent the system state and the offloading decision is used to define the action. The solution is then derived using an efficient distributed deep reinforcement-learning-based algorithm. In conclusion, experimental results demonstrate that our model converges quickly and significantly reduces system cost (i.e., about 41.9%, 44.2%, and 11%) compared to local execution, full offloading policies, and the task offloading work in [47].

In future work, the proposed model will be extended and we address the limitations of bandwidth, especially for larger data size transmissions, by introducing a new data compression layer. Moreover, protecting the applications' data during the offloading process will be addressed. Finally, a more general case of multi-users and multiple edges of the MEC system will be considered, in which the mobility between edge servers is considered.

Author Contributions: Conceptualization, I.A.E. and S.M.; methodology, I.A.E.; software, M.H.; validation, S.M. and I.A.E.; formal analysis, I.A.E. and S.M.; investigation, I.A.E.; resources, S.M.; data curation, M.H.; writing—original draft preparation, I.A.E.; writing—review and editing, S.M. and M.H.; visualization, I.A.E.; supervision, S.M.; project administration, I.A.E.; funding acquisition, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Princess Nourah Bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R196), Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge the Princess Nourah Bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R196), Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Asghari, P.; Rahmani, A.M.; Javadi, H.H.S. Internet of Things applications: A systematic review. *Comput. Netw.* **2019**, *148*, 241–261. [CrossRef]
2. Paramonov, A.; Muthanna, A.; Aboulola, O.I.; Elgendy, I.A.; Alharbey, R.; Tonkikh, E.; Koucheryavy, A. Beyond 5g network architecture study: Fractal properties of access network. *Appl. Sci.* **2020**, *10*, 7191. [CrossRef]
3. Fagroud, F.Z.; Ajallouda, L.; Lahmar, E.H.B.; Toumi, H.; Zellou, A.; El Filali, S. A Brief Survey on Internet of Things (IoT). In Proceedings of the International Conference on Digital Technologies and Applications, Fez, Morocco, 29–30 January 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 335–344.
4. Kim, Y.G.; Kong, J.; Chung, S.W. A survey on recent OS-level energy management techniques for mobile processing units. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 2388–2401. [CrossRef]
5. Kumar, K.; Liu, J.; Lu, Y.H.; Bhargava, B. A survey of computation offloading for mobile systems. *Mob. Netw. Appl.* **2013**, *18*, 129–140. [CrossRef]
6. Khayyat, M.; Elgendy, I.A.; Muthanna, A.; Alshahrani, A.S.; Alharbi, S.; Koucheryavy, A. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access* **2020**, *8*, 137052–137062. [CrossRef]
7. Kalra, V.; Rahi, S.; Tanwar, P.; Sharma, M.S.; et al. A Tour Towards the Security Issues of Mobile Cloud Computing: A Survey. In *Emerging Technologies for Computing, Communication and Smart Cities*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 577–589.
8. Chakraborty, A.; Mukherjee, A.; Bhattacharyya, S.; Singh, S.K.; De, D. Multi-criterial Offloading Decision Making in Green Mobile Cloud Computing. In *Green Mobile Cloud Computing*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 71–105.
9. Othman, M.; Madani, S.A.; Khan, S.U. A survey of mobile cloud computing application models. *IEEE Commun. Surv. Tutorials* **2013**, *16*, 393–413.
10. Noor, T.H.; Zeadally, S.; Alfazi, A.; Sheng, Q.Z. Mobile cloud computing: Challenges and future research directions. *J. Netw. Comput. Appl.* **2018**, *115*, 70–85. [CrossRef]
11. Elgendy, I.A.; Zhang, W.Z.; Liu, C.Y.; Hsu, C.H. An efficient and secured framework for mobile cloud computing. *IEEE Trans. Cloud Comput.* **2018**, *9*, 79–87. [CrossRef]
12. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
13. Elgendy, I.A.; Yadav, R. Survey on mobile edge-cloud computing: A taxonomy on computation offloading approaches. In *Security and Privacy Preserving for IoT and 5G Networks*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 117–158.
14. Zhang, S.; Zhang, H.; Di, B.; Song, L. Joint trajectory and power optimization for UAV sensing over cellular networks. *IEEE Commun. Lett.* **2018**, *22*, 2382–2385. [CrossRef]
15. Alzenad, M.; El-Keyi, A.; Lagum, F.; Yanikomeroglu, H. 3-D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 434–437. [CrossRef]

16. Motlagh, N.H.; Bagaa, M.; Taleb, T. UAV-based IoT platform: A crowd surveillance use case. *IEEE Commun. Mag.* **2017**, *55*, 128–134. [CrossRef]
17. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2322–2358. [CrossRef]
18. Liu, L.; Chang, Z.; Guo, X.; Ristaniemi, T. Multi-objective optimization for computation offloading in mobile-edge computing. In Proceedings of the 2017 IEEE symposium on computers and communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 832–837.
19. Alhelaly, S.; Muthanna, A.; Elgendy, I.A. Optimizing Task Offloading Energy in Multi-User Multi-UAV-Enabled Mobile Edge-Cloud Computing Systems. *Appl. Sci.* **2022**, *12*, 6566. [CrossRef]
20. Pham, Q.V.; Fang, F.; Ha, V.N.; Piran, M.J.; Le, M.; Le, L.B.; Hwang, W.J.; Ding, Z. A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access* **2020**, *8*, 116974–117017. [CrossRef]
21. Chen, J.; Chen, S.; Luo, S.; Wang, Q.; Cao, B.; Li, X. An intelligent task offloading algorithm (iTOA) for UAV edge computing network. *Digit. Commun. Netw.* **2020**, *6*, 433–443. [CrossRef]
22. Qi, W.; Sun, H.; Yu, L.; Xiao, S.; Jiang, H. Task Offloading Strategy Based on Mobile Edge Computing in UAV Network. *Entropy* **2022**, *24*, 736. [CrossRef]
23. Yu, Z.; Gong, Y.; Gong, S.; Guo, Y. Joint task offloading and resource allocation in UAV-enabled mobile edge computing. *IEEE Internet Things J.* **2020**, *7*, 3147–3159. [CrossRef]
24. Zhu, S.; Gui, L.; Zhao, D.; Cheng, N.; Zhang, Q.; Lang, X. Learning-based computation offloading approaches in UAVs-assisted edge computing. *IEEE Trans. Veh. Technol.* **2021**, *70*, 928–944. [CrossRef]
25. Zhao, N.; Ye, Z.; Pei, Y.; Liang, Y.C.; Niyato, D. Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-assisted Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 6949–6960. [CrossRef]
26. Sacco, A.; Esposito, F.; Marchetto, G.; Montuschi, P. Sustainable task offloading in UAV networks via multi-agent reinforcement learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5003–5015. [CrossRef]
27. He, X.; Jin, R.; Dai, H. Multi-hop task offloading with on-the-fly computation for multi-UAV remote edge computing. *IEEE Trans. Commun.* **2021**, *70*, 1332–1344. [CrossRef]
28. Luo, Y.; Ding, W.; Zhang, B. Optimization of task scheduling and dynamic service strategy for multi-UAV-enabled mobile-edge computing system. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 970–984. [CrossRef]
29. Yang, L.; Yao, H.; Wang, J.; Jiang, C.; Benslimane, A.; Liu, Y. Multi-UAV-enabled load-balance mobile-edge computing for IoT networks. *IEEE Internet Things J.* **2020**, *7*, 6898–6908. [CrossRef]
30. Li, W.T.; Zhao, M.; Wu, Y.H.; Yu, J.J.; Bao, L.Y.; Yang, H.; Liu, D. Collaborative offloading for UAV-enabled time-sensitive MEC networks. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 1–17. [CrossRef]
31. Zhou, Y.; Ge, H.; Ma, B.; Zhang, S.; Huang, J. Collaborative task offloading and resource allocation with hybrid energy supply for UAV-assisted multi-clouds. *J. Cloud Comput.* **2022**, *11*, 1–13. [CrossRef]
32. He, Y.; Zhai, D.; Huang, F.; Wang, D.; Tang, X.; Zhang, R. Joint task offloading, resource allocation, and security assurance for mobile edge computing-enabled UAV-assisted VANETs. *Remote Sens.* **2021**, *13*, 1547. [CrossRef]
33. Munawar, S.; Ali, Z.; Waqas, M.; Tu, S.; Hassan, S.A.; Abbas, G. Cooperative Computational Offloading in Mobile Edge Computing for Vehicles: A Model-based DNN Approach. *IEEE Trans. Veh. Technol.* **2022**, 1–16. [CrossRef]
34. Mohamed, H.; Al-Masri, E.; Kotevska, O.; Souri, A. A Multi-Objective Approach for Optimizing Edge-Based Resource Allocation Using TOPSIS. *Electronics* **2022**, *11*, 2888. [CrossRef]
35. Chen, J.; Cao, X.; Yang, P.; Xiao, M.; Ren, S.; Zhao, Z.; Wu, D.O. Deep Reinforcement Learning Based Resource Allocation in Multi-UAV-Aided MEC Networks. *IEEE Trans. Commun.* **2022**, *71*, 296–309. [CrossRef]
36. Xu, D.; Xu, D. Cooperative task offloading and resource allocation for UAV-enabled mobile edge computing systems. *Comput. Netw.* **2023**, *223*, 109574. [CrossRef]
37. Chai, F.; Zhang, Q.; Yao, H.; Xin, X.; Gao, R.; Guizani, M. Joint Multi-task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT. *IEEE Trans. Veh. Technol.* **2023**, 1–15. [CrossRef]
38. Banerjee, A.; Sufian, A.; Paul, K.K.; Gupta, S.K. Edtp: Energy and delay optimized trajectory planning for uav-iot environment. *Comput. Netw.* **2022**, *202*, 108623. [CrossRef]
39. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* **2015**, *24*, 2795–2808. [CrossRef]
40. Zhang, W.Z.; Elgendy, I.A.; Hammad, M.; Iliyasu, A.M.; Du, X.; Guizani, M.; Abd El-Latif, A.A. Secure and optimized load balancing for multitier IoT and edge-cloud computing systems. *IEEE Internet Things J.* **2020**, *8*, 8119–8132. [CrossRef]
41. Elgendy, I.A.; Zhang, W.; Tian, Y.C.; Li, K. Resource allocation and computation offloading with data security for mobile edge computing. *Future Gener. Comput. Syst.* **2019**, *100*, 531–541. [CrossRef]
42. Deb, S.; Monogioudis, P. Learning-based uplink interference management in 4G LTE cellular systems. *IEEE/ACM Trans. Netw.* **2014**, *23*, 398–411. [CrossRef]
43. Dinh, T.Q.; Tang, J.; La, Q.D.; Quek, T.Q. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Trans. Commun.* **2017**, *65*, 3571–3584.
44. Fooladivanda, D.; Rosenberg, C. Joint resource allocation and user association for heterogeneous wireless cellular networks. *IEEE Trans. Wirel. Commun.* **2012**, *12*, 248–257. [CrossRef]

45. Ong, H.Y.; Chavez, K.; Hong, A. Distributed deep Q-learning. *arXiv* **2015**, arXiv:1508.04186.
46. Chen, M.H.; Liang, B.; Dong, M. Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
47. Huang, L.; Feng, X.; Zhang, L.; Qian, L.; Wu, Y. Multi-server multi-user multi-task computation offloading for mobile edge computing networks. *Sensors* **2019**, *19*, 1446. [CrossRef] [PubMed]
48. Almutairi, J.; Aldossary, M.; Alharbi, H.A.; Yosuf, B.A.; Elmirghani, J.M. Delay-Optimal Task Offloading for UAV-Enabled Edge-Cloud Computing Systems. *IEEE Access* **2022**, *10*, 51575–51586. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.