Received July 20, 2021, accepted August 4, 2021, date of publication August 11, 2021, date of current version August 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3104117

Fast and Fair Computation Offloading Management in a Swarm of Drones Using a Rating-Based Federated Learning Approach

DADMEHR RAHBARI[1], MUHAMMAD MAHTAB ALAM[1], (Senior Member, IEEE), YANNICK LE MOULLEC[1], (Senior Member, IEEE), AND MAKSIM JENIHHIN[2], (Member, IEEE)

[1]Thomas Johann Seebeck Department of Electronics, School of Information Technologies, Tallinn University of Technology, 19086 Tallinn, Estonia
[2]Department of Computer Systems, Tallinn University of Technology, 12618 Tallinn, Estonia

Corresponding author: Dadmehr Rahbari (dadmehr.rahbari@taltech.ee)

This work in the project ''Information and Communication Technologies (ICT) programme'' was supported by the European Union through European Social Fund. This work was also supported by the European Commission under Horizon 2020 European Reseach Aera (ERA)-chair Grant 'COgnitive ELectronics (COEL)' (Agreement number: 668995), Tallinn University of Technology Development Program 2016-2022 (2014-2020.4.01.16-0032) and Estonian Centre of Excellence in ICT Research (EXCITE).

ABSTRACT Today, unmanned aerial vehicles (UAVs) or drones are increasingly used to enable and support multi-access edge computing (MEC). However, transferring data between nodes in such dynamic networks implies considerable latency and energy consumption, which are significant issues for practical real-time applications. In this paper, we consider an autonomous swarm of heterogeneous drones. This is a general architecture that can be used for applications that need in-field computation, e.g. real-time object detection in video streams. Collaborative computing in a swarm of drones has the potential to improve resource utilization in a real-time application i.e., each drone can execute computations locally or offload them to other drones. In such an approach, drones need to compete for using each other's resources; therefore, efficient orchestration of the communication and offloading at the swarm level is essential. The main problem investigated in this work is computation offloading between drones in a swarm. To tackle this problem, we propose a novel federated learning (FL)-based fast and fair offloading strategy with a rating method. Our simulation results demonstrate the effectiveness of the proposed strategy over other existing methods and architectures with average improvements of -23% in energy consumption, -15% in latency, $+18\%$ in throughput, and $+9\%$ in fairness.

INDEX TERMS Swarm of drones, multi-access edge computing, collaborative computing, federated learning.

I. INTRODUCTION

Autonomous swarms of unmanned aerial vehicles (UAVs) or drones have attracted significant attention in different application domains. When operating in a group, drones can process the computation tasks independently or collaboratively. A group of independent drones is subject to severe constraints such as limited battery capacity, computing resource capability, etc. On the other hand, **collaboration** between drones, i.e. offloading computations between them, yields many advantages over a group of independent drones [1]. Indeed, a swarm of collaborative and intelligent drones decreases the overall energy consumption and cost, and increases the throughput and quality of service (QoS) [2]. Generally, a swarm of drones can be used for many applications such as monitoring, detection, etc. [3], [4].

In a **swarm**, drones can execute computation tasks locally or offload them to other devices if doing so is more efficient at the swarm level. The scientific literature shows that considering the cloud as a centralized system is not always a suitable

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

option for offloading, taking into account latency, QoS, and energy consumption constraints. Moreover, there are some environmental limitations for deploying datacenters in reliable positions. To overcome these problems, **multi-access edge computing (MEC)** can be a more suitable architecture than the cloud [5], [6]. Edge servers are closer to drones and the communication is faster than through a cloud [7], [8]; however, the problems of latency and energy consumption remain. Alternatively, drones can be on the edge and process the received data from external [9] or internal sensors. The efficiency of the latter architecture strongly depends on an efficient resource allocation strategy [10].

Collaborative computing of drones can be organized in centralized or distributed architectures. A **centralized architecture** can include the drones, edge servers, cloud, etc., where all computations are finalized in a server [6]. In case the drones receive a large volume of data, there will be many tasks to be executed. Since the resources of servers are limited, we face a queue of tasks in each device. In the centralized architecture, some tasks will be dropped; indeed, when drones or the centralized server are busy and cannot service tasks as soon as they enters the queue, the tasks have to wait or be dropped. On the other hand, in a **distributed architecture**, since each device can have a comprehensive view of the network, a given device can execute the computation locally or offload it to other devices in the network. Yet, the swarm of drones and communication between them without any centralized node face some challenges such as latency and energy consumption [11]. The processing capacity and power of drones is lower than that of servers, which means that providing appropriate and optimal methods for resource management (orchestration) in a swarm of drones is highly needed. A distributed resource orchestration is preferable to a centralized one due to offloading some tasks to other devices [12]. In fact, a distributed architecture decreases the number of delayed and dropped tasks.

In our work, a task refers to a module (a component of a software application) that includes some operations and data. The computation **task offloading** process means transferring the task to other nodes if the resources or performance of the current node is not sufficient [6]. There are different meanings of **distributed offloading**. On the one hand, a distributed resource management method can offload tasks with their data to other devices. In this way, due to transferring data among several devices, data privacy is a serious challenge [13]. On the other hand, a distributed algorithm can decide the destination of tasks for offloading without transferring any data of the tasks and only transmit the learning model [14]. In our work, we consider the second meaning. Given this, the main problem of this work is finding **the best destination of tasks for offloading by drones**. This takes place by means of a distributed strategy in a swarm of drones. From an application perspective, the main challenge addressed in this paper is to decrease the latency and energy consumption for the execution of the mission-required computations in the swarm of drones. Here, the latency means delays in data transfer between drones, as well as delays in the execution of tasks.

Machine learning (ML) algorithms are known to be successfully applied in intelligent solutions for resource management or computing problems with the mentioned issues [15]. Among these, **federated learning (FL)** is a distributed strategy that has been used for MEC architecture [16]. The FL spreads and also aggregates a learning model over the network. Deep reinforcement learning (DRL) as a high performance ML algorithm has been used in drone applications; e.g. in [17], an extension of DRL with a convolutional neural network (CNN) and transfer learning used for accelerating the learning process has been proposed. However, this is not a distributed strategy, because the algorithm is executed in one device and not in cooperation with other devices. In this paper, we propose a distributed DRL for the offloading process in the swarm of drones.

DRL is a conventional reinforcement learning (RL) method. RL considers the problem of a computational agent that makes decisions using trial and error [18]. DRL includes in-depth learning in solutions, allowing agents to make decisions from unstructured input data without manually engineering the solution space [19]. Q-learning (QL) is a model-free RL algorithm to learn the quality of actions [20]. There exist extensions of this algorithm such as double q-learning (DQL) [21]; the difference between DQL and Q-learning is in the evaluation of actions. In the scientific literature, FL has been combined with DRL; the resulting federated deep reinforcement learning (FDRL) is a distributed version of DRL that can be used in MEC [22]. FDRL highly reduces the network overhead, bandwidth consumption, and latency by avoiding sending data towards a central entity. FDRL is a parallel scheme that learns the whole of the networks such as the properties of drones (e.g. energy consumption, latency, bandwidth, and CPU frequency) and the best drones in the swarm for offloading the modules. Since drones cannot independently and efficiently support centralized schemes of DRL due to the above-mentioned restrictions, the **FDRL** concept is more suitable for a swarm of drones based on wireless [23] and 5G networks [24].

One of the **challenges of FL** is aggregating learning model weights. Let's assume several drones are tracking an object. It is possible that some of them do not have sufficient resources and also that they are not in the right position or have an unsuitable camera angle. The existing FL-based strategies aggregate all of the learning models [14], [25]. This builds some weakness in the offloading processes as follows. First, due to the lack of drones in the right position [26], proper data is not tracked and as a result, the local learning algorithm cannot set the best weights. Second, drones that do not have sufficient computational power to process also participate in the aggregation operation, which will lead to inadequate learning. To solve this problem, we propose to rate the drones with their mentioned properties, then we can update the model based on aggregating drones' models with the higher rating values. This method has two advantages, the first one is to build a suitable and efficient learning model and the second one is to update the position of drones based on the ratings received from their neighbors. In the following, we explain the main objective of this work, i.e. computation offloading management and our strategy to solve it.

To address the above issues, this paper proposes a novel FL strategy based on DQL and the rating mechanism of drones named FLR in MEC architecture. Drones can execute related computation tasks locally or offload that to other drones with FL-based cooperation in a distributed environment. FL helps drones to communicate together without worrying about data privacy and improving the latency and energy consumption. In fact, the learning model weights are transferred within the swarm. Since each drone receives many weights, we consider a rating method to gain the best learning model and spread it to neighbors. For the distributed offloading and keeping a balance between drones, a **fairness** value is calculated based on the energy and latency of drones.

The main contributions of this work are as follows. We propose a novel distributed offloading management strategy that is based on online FDRL, which also relies on drone rating. The rating is computed and constantly (on offloading events) re-computed for each drone, based on a set of its current properties reflecting a drone's potential to execute the received task. Such task execution is subject to constraints of required performance in terms of key parameters such as energy consumption, latency, throughput, and fairness. We consider a collaborative edge computing architecture in a dynamic network formed by an autonomous swarm of drones. An example of the application for this setup is real-time object detection and/or tracking. We specifically include the heterogeneity of the drones and their respective configurations (e.g., computational power and communication bandwidth resources) in our system model. We compare the performance of the proposed approach against that of well-known benchmarks (random, greedy, and hierarchical offloading methods). A comprehensive evaluation is performed based on the number of drones. Our results show that the proposed approach decreases the energy consumption by 23%, latency in both communication and computation by 15%, and increases the throughput by 18% and fairness by 9%, on average, as compared to the benchmarks methods.

This paper is organized as follows. First, a survey of FL related works and offloading methods in MEC is presented in Section II. In Section III, we explain the architecture and challenges for module offloading in MEC-based swarm of drones. Then, the system model is presented in Section IV. Next, in Section V, we present the FL-based offloading framework in swarm of drones. After that in Section VI, the performance evaluation demonstrates the effectiveness of the proposed approach. Finally, in Section VII, a conclusion and suggestions for future work are provided.

II. RELATED WORK

In this section, we first (in Subsection II-A) describe resource management works in a group of drones in different architectures including the cloud, edge, and IoT devices. Then, we explain the concept of FL and related research in MEC (in Subsection II-B).

A. RESOURCE MANAGEMENT IN A GROUP OF DRONES IN DIFFERENT ARCHITECTURES

Here, we present the related works in resource management which have used drones as end devices, edge devices, or edge servers that work independently or collaboratively.

There exist works on resource management with different architectures including {cloud, edge servers, and drones [5], [7], [27]}, {mobile users, drones [28]}, {wireless users, edge servers, and drones [29]}, {drones and edge servers [30], [31]}, and {drones, edge servers, and smart mobile devices [32], [33]}.

A **swarm of drones** has also been used in [34] for the offloading and pricing problems. That research was simulated based on a number of blockchain **servers**, **edge servers**, **drones**, and **IoT devices**. The method was based on an unsupervised hierarchical RL, Bayesian learning, and DL for a stochastic Stackelberg game with multiple leaders under incomplete information. An objective function as the payoff was built using delay, energy, and cost of communication and computation. However, time complexity analysis was not provided.

Another type of architecture is said to be flat and includes a **group of drones** without any centralized servers. Researchers in [13] presented a distributed algorithm based on proximal Jacobi alternating direction method of multipliers (ADMM) to solve the Fog-Cloud-based computation offloading problem in a swarm of drones. They considered latency, reliability, and energy consumption. The results showed the low computational complexity and reliability of the proposed approach compared to the genetic algorithm (GA), centralized Jacobi ADMM, and random offloading.

Authors in [35]–[37] worked on optimizing the energy consumption in a **swarm of drones**. In [35], the drones are classified into two types, i.e., user devices (UDs) and helper devices (HDs) where HDs have more computing capability than UDs. There is also a cluster head for gathering and updating a learning automata for computation offloading. Their method optimized the energy consumption in comparison with locally, random, and traversing algorithms. However, they did not consider what happens to the computation tasks in the swarm when the cluster head (i.e. the master node) becomes unavailable. In [36], the authors also applied two classes of drones. Considering their respective computation capability, one type was used for detection and the other one for computation. There are specific pairs for each drone for offloading the computations. They modeled the problem with game theory by generalized Nash equilibrium strategy. There was only one drone for updating the learning model. In [37], the deployment and computation problem is solved by a random best and better response (RAN-BBR) algorithm. The authors clustered drones into some coalitions with the same altitude. There was also a cluster head in each coalition.

Their method is randomly executed in only one of the coalitions at each repetition and is not distributed.

Obviously, the presented system model and learning strategies in [35]–[37] are not fully distributed and the computation offloading in each cluster also depends on the decision by the cluster head, which limits those works. In fact, each time only one drone is randomly aggregating the learning model and the others do not have any aggregation at the same time, i.e. it is a master and slave architecture. On the contrary, in our proposed approach, all drones are updating and aggregating their learning model. This has the advantage that all drones in a swarm are aware of the status of the whole network to make an efficient decision about local computation or offloading.

B. FEDERATED LEARNING FOR RESOURCE MANAGEMENT

Here, a definition of FL and its steps are provided. Then, we discuss some related works in FL for solving the resource management problems.

1) DEFINITION OF FL

FL allows devices to collaboratively train a global model regarding the privacy of data and improve the latency and energy consumption in all devices. This means FL has a distributed structure. As a result, FL can enable ML model training with MEC. In its basic concept, FL includes some data owners denoted $\{device_1, device_2, \ldots, device_N\}$ and the model owner (FL server). There are three main steps in FL:

1) Step 1 takes place in the FL server, including the initialization of the hyperparameters such as the global model and the training process, e.g., learning rate.
2) In step 2, each $device_i$ trains a model's w_i locally and transfers it to the FL server.
3) In step 3, all collected local models in the FL server are aggregated by

$$W = \frac{1}{N} \sum_{i=1}^{N} w_i \tag{1}$$

where W is weight of global learning model and w_i are weights of learning model i.

Steps 2 and 3 are repeated until completed or until a desired training accuracy is reached [16]. Using FL in MEC has the following advantages: i) less data is required to be transmitted to a central server; ii) no raw data and only the updated model parameters are transmitted between devices for aggregation; iii) FL enhances data privacy when the raw data of users need not be transferred or sent to another device; iv) since tasks can be executed locally, the latency is decreased.

2) FL-BASED METHODS FOR RESOURCE MANAGEMENT

Some research efforts dealt with improving the FL steps. Authors in [38] classified some of them. These improvements have been applied on training tricks [39], client selection [16], [40], data compensation [41], hierarchical aggregation [42], model compression [43], knowledge distillation [44], and asynchronous update [45], [46]. The above works show that in some implementations, the FL servers for synchronization of the learning models can be replaced with regular nodes.

In recent years, FL has been used in drones' applications such as wireless signal propagation, trajectory planning, deployment and placement, content caching, and data routing in a multi-hop manner [23]. Typically, drones were participants and an edge server or a cloud was an aggregator of FL. In our paper, we propose the FL for finding the best destination for offloading the tasks. Since our problem is distributed offloading in a swarm of drones as MEC, we present the works related to cooperative computing in the following subsections.

FL has been used for drones in different architectures. In [47], a group of researchers provided a joint Auction-Coalition technique for centralized FL. Their model included some roadside units (RSU) as FL participants, i.e. vehicles, and a cloud as owner FL and some drones. In that network, drones were responsible for sending FL worker trained model to the owner. They analyzed their proposed approach with energy consumption and communication time. However, their work was not compared with any other research. The work presented in [48] was based on an Internet of vehicles (IoV) system controlled by FL. That work is based on a centralized FL server, some drones with local FL, and also some vehicles. According to that research, the lowest cost drones are matched to sub-regions, and then sensing and local FL training are performed. One of the issues here is that drones are dependent on a centralized system and without it, the learning model cannot be updated. This work was analyzed by profit and utility of drones without any comparison with other methods.

Another architecture involves one or several drones and edge servers in two layers of the network. In [14], an FL method based on convolution neural networks and DL was provided for image classification tasks to reduce the communication cost between the drones and the ground fusion center (GFC). The results showed that FL has higher accuracy with a lower cost than the centralized methods. However, that work ignored latency and energy consumption in communication and computation. The authors in [49] presented a battery-constraint FL in drones with a centralized edge server. They aimed to optimize the loss function in DRL for improving the latency and energy consumption. However, the moving of drones and their distances from edge server were ignored.

Table 1 classifies the related works based on the network architecture (Arc.), objectives, technique, environment, pros, and cons, and whether they use distributed resource management (DRM). The study of the related works shows that on one hand, most researchers used a swarm of drones as a section of a larger architecture. Some of them aimed to address the computation or communication challenges or both of them. Interestingly, many of the works, despite having a distributed architecture, do not use a distributed resource management method and do not use the ability to interact between devices. In fact, these works leave the decision about

TABLE 1. Overview of distributed offloading methods.

Ref	Arc.	Objectives	Technique	Environment	Pros	Cons	DRM
[13]	Drones	Latency, reliability, and energy consumption	Jacobi ADMM	Simulation	Considers communication and computation latency and energy consumption. Low computation complexity.	Ignores fault tolerant model for reliability. Lack of evaluation for many drones.	✓
[14]	Drones/ GFC	Accuracy and cost	FL	Simulation	More accuracy in image classification and lower cost.	Ignores energy consumption and latency.	✓
[34]	Drones/ MESs/ BS/IoT	Delay, cost, energy consumption	Hybrid RL, DQL, Bayesian, Markov	OPNET	Online processing without any offline training.	High time complexity. Ignores detailed analysis of delay, cost, and energy consumption.	✓
[49]	Drones/ MESs	Latency, energy consumption	FL	Pytorch	Distributed resource management.	Ignores the moving of drones.	✓
[35]	Drones	Energy consumption	Learning automata	Simulation	Decreasing running time and energy consumption	Offloading is dependent on a cluster head	✓
[36]	Drones	Energy consumption	Game theory	Simulation	Improving energy consumption	Existing one model updater. Not fully distributed method.	
[37]	Drones	Energy consumption	RAN-BBR	Simulation	Faster convergence speed	No 3D movement. Not fully distributed method.	
[30]	Drones/ MES	Delay and energy consumption	DL	TensorFlow	Improvement of object detection in bad quality images.	Evaluated in a very small network. No comparison with other works.	
[31]	Drones/ MESs	Energy consumption and latency	GA	Real	Considers communication and computation time and energy consumption	Time consuming method. No comparison with other works.	
[32]	Drones/ MESs/ SMDs	Energy consumption	Approximation technique	Simulation	Improved energy consumption than SCA method.	No scalablity. High time complexity.	
[33]	Drones/ MESs/ SMDs	Computation efficiency and energy consumption	Dinkelbach algorithm	Simulation	Improves computation efficiency and energy consumption	Ignores computation and communication latency.	
[29]	Drones/ MESs/ WUs	delay, energy consumption, and bandwidth cost	DRL	Python/ Tensorflow	Uses solar generator system. Maximizes rewards of DRL.	No detailed analysis of objectives.	
[7]	Drones/ MESs/ Cloud	Migration cost	Markov approximation and Lyapunov	Gurobi	Lower migration cost than standard Markov-based method.	Ignores energy consumption and latency.	
[5]	Drones/ MESs/ Cloud	QoS and cost	Markov approximation	Python	High latency and power consumption	No comparison with other methods.	
[48]	Drones/ RSU/ Cloud	Profit and utility	FL/Multi-Dimensional Contract-Matching	Simulation	Drones-based sub region formulation.	Drones dependence on a centralized system. No comparison with other works.	
[47]	Drones/ RSU/ Cloud	Energy consumption and communication time	FL with auction-coalition	Simulation	Considers communication, flying, hovering, and computation energy consumption.	Ignores the computation latency. No comparison with other methods.	

Note: Arc (Architecture), MES (Mobile edge server), and DRM (Distributed resource management).

the best resource management operations to a central server such as a cloud or multiple edge servers. This is while the drones themselves have the ability to process and decide on this. On the other hand, these works indicate that working on FL steps like the selection of learning model from devices and aggregation model have some advantages but also pose a number of challenges. Therefore, in our paper, we work to fill this research gap, that is, to use the ability to interact between drones with comprehensive learning throughout the network, so that the challenges related to communication and computation are also applied.

III. COLLABORATIVE EDGE COMPUTING ARCHITECTURE IN A SWARM OF DRONES

As shown in Figure 1, in the assumed architecture, there are only some drones without any edge server or cloud. We consider, as an example application, that drones detect in real time objects in the captured video streams. On the one hand, these operations are computationally demanding; on the other hand, due to the heterogeneity of drones, their

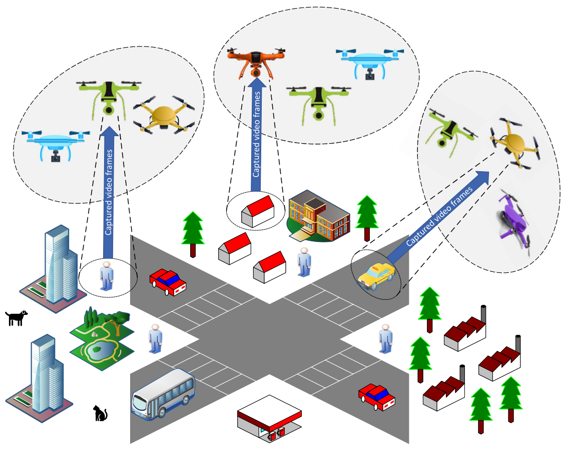

FIGURE 1. Collaborative edge computing architecture in swarm of drones.

respective computing capacity are different. This means that some drones with high capacity can execute the process locally and some others have to send their computation workload to other drones.

Offloading modules to close-by neighbors can yield lower latency, energy consumption, and privacy risk. Our objective is to find the best destination for drones to offloading their computations.

Some drones may not be able to perform tasks at a given time due to insufficient battery, processing capacity, or real-time constraints and may have to request their neighbors to perform these tasks. FLR is a distributed process suitable for the problem of managing resource allocation. Yet, it comes with challenges as outlined below.

A. CHALLENGES AND SOLUTIONS OF TASK OFFLOADING IN DISTRIBUTED DRONES IN MEC

1) LATENCY

Slow and unstable communication is a serious problem. If the drones send their modules to far-away drones in the network, latency increases. However, clustering of drones helps them to perform offloading to closer neighbor drones. In the proposed approach, drones are not dependent on any edge servers. As some nodes become offline, only the weights are less trained or updated later, but the task performing or offloading is done continuously, at least until there are enough drones alive in the swarm.

2) ENERGY CONSUMPTION

Transferring data between drones increases energy consumption. FL helps alleviating this problem. In FL, drone data are not sent to other drones and only weights' of the learning models are transferred; thus, less energy is needed. Another energy challenge is that drone's switching off may affect the learning performance. This can be solved by receiving drones' properties from neighbors. The drones know each other's characteristics and this is used in calculating rankings. As a result, having this information in each device also helps in offloading operations.

3) COMPLEXITY OF THE MODEL

Comparisons with other distributed learning methods show that complexity is an important challenge. Some methods use neural networks up to a cut layer and others ignore communication weights to an aggregating server. FL is easier to implement since the participants and the FL server are running the same global model in each cluster. Moreover, drones may be physically close to each other. This introduces a collision issue when they update local models to the neighbors. As such, channel allocation policies may need to be combined with the resource allocation approaches to address the collision issue. This means drones can predict the best trajectory (see the outlook for future work in Section VII) for optimizing sending their modules to others. Since FL provides a global view of the network for all drones, they can thus control their movement to avoid collision and find the best neighbor for offloading. DQL can be considered to model the dynamic environment of MEC and make optimized decisions. As a solution, drone properties such as CPU frequency, bandwidth, execution time, and energy consumption can be used for presenting an objective function for DQL. In fact, this learning algorithm finds the optimal destination for offloading the modules. The size of model updates can be another challenge. However, the high-speed communication in modern technologies such as 5G helps transferring large data fast.

IV. SYSTEM MODEL

FL can be used for addressing the above-mentioned issue by training a shared global model from a swarm of drones. In this paper, we propose a new module-offloading framework employing FL for decision making to find the best destination for offloading. Each drone becomes aware of the drones around it without having to send its data to a central system such as a cloud or a semi-centralized system such as fixed edge servers to gain this knowledge.

Figure 2 presents the framework of module offloading between drones by means of the proposed FL strategy. This is a sequence diagram that shows what happens in each drone and its neighbors in a swarm. The main contribution of this work is the rating of neighbors and updating the learning model with aggregating the higher-rated models, as

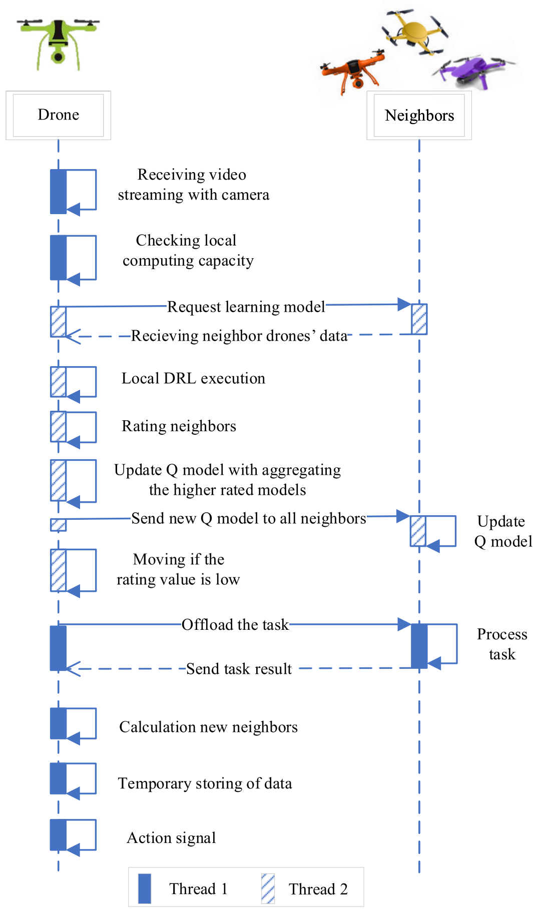
Drone
Neighbors
Receiving video streaming with camera
Checking local computing capacity
Request learning model
Recieving neighbor drones' data
Local DRL execution
Rating neighbors
Update Q model with aggregating the higher rated models
Send new Q model to all neighbors
Update Q model
Moving if the rating value is low
Offload the task
Process task
Send task result
Calculation new neighbors
Temporary storing of data
Action signal
Thread 1
Thread 2

FIGURE 2. Framework of FLR for offloading in a Swarm of drones.

highlighted by the green text. Moreover, there are two running threads in each drone that execute instructions on different processors simultaneously.

All terms used in this work are presented in Table 2.

A. SWARM OF DRONES

We consider multiple drones, each denoted as D_i where $i = \{1, 2, \ldots, I\}$. The properties of each drone include position (X, Y, Z), CPU frequency, amount of RAM, communication bandwidth, idle power, and active power. To mimic the heterogeneity of the drones in the swarm, at first, a certain value of these parameters is considered and these sets of values are then multiplied by one of the numbers in the set $\{Drone; Type_1 = 1, Type_2 = 1.5, Type_3 = 2, Type_4 = 2.5, Type_5 = 3\}$. In other words, the values 1, 1.5, 2, etc. are multiplying factors applied to the parameters' values of the first drone, in order to generate more drones' configurations with different values. For simulation purposes, we consider 5 types of drones in the environment. For example, for a swarm of 50 drones, we may have 10 drones of each type.

Assumption 1: All drones are in the air. This means we assume that the primary position of the drones is the air, thus they are not in the idle, take off, armed, and payload modes. They can be in other states such as hovering and flying.

Assumption 2: The camera's angle of all drones is fixed. Since the problem of this research is application module offloading, setting the camera's angle and finding the best direction thereof in drones is not this research's purpose. Here, we assume that if an object is in the camera coverage of a drone then the drone can track the object. However, we consider distance between drones and objects to calculate the performance value.

B. APPLICATION SCENARIO

An example of the application assumed for the objective of the proposed analysis is object detection by drones based on video streams sensed in real time. An application includes several modules. Each module involves the data processing elements. Data generated as output by each module may be used as input by another module. This application model creates a directed graph, with the vertices representing application modules and directed edges showing the data flow between modules [50].

Figure 3 illustrates possible involved embedded software modules i.e. sensing, client, processing, storage, and actuation. The sensing module can send camera video streams to the client module. The client module executes pre-processing of the sensor-generated data and transfers it to the processing module. The processing module provides two types of data, i.e. processed and action commands, that can be transferred to storage and client modules, respectively. The storage module can be used for saving of processing modules for location-independent and scalable distribution. Moreover, the client module can generate the actuation signal.

The above modules form an application for tracking objects according to a workflow. For more clarification, it should be noted that the main objective of this research is the management of computation offloading between drones in a swarm, and we do not contribute to a solution for image

TABLE 2. Term definition.

Term	Definition
DRL	Deep Reinforcement Learning
DQL	Deep Q Learning
FDRL	Federated Deep Reinforcement Learning
FLR	Federated Learning with Rating in a swarm of drones
Hierarchical	FLR offloading between some drones and edge servers
Random	Random offloading in a swarm of drones
Greedy	Greedy offloading in a swarm of drones
D_i	Drone i
N^D	Number of drones
O_k	Object k
L_i^{Loc}	Local latency of drone i
E_i^{Loc}	Local computation energy consumption of drone i
E_i^{Comp}	Computation energy consumption of drone i
E_i^{Tra}	Transmitting energy consumption of drone i
E_i^{Rec}	Receiving energy consumption in offloading from drone i
E_i^{H}	Hovering energy consumption of drone i
E_i^{FH}	Flying Horizontally energy consumption of drone i
E_i^{FVU}	Flying Vertically Upward energy consumption of drone i
E_i^{FVB}	Flying Vertically Backward energy consumption of drone i
E_i^{Mov}	Moving energy consumption of drone i
E_i^{Tot}	Total energy consumption of drone i
f_i and f_j	CPU frequency of drones i and j
β	Rate of communication size
γ	Path loss exponent
ω_m	Size of module m
η	Drones' moving rate
$R^{\text{DL}}(i, j)$	Distance-based link rate between drone i to drone j
d_{ij}	Distance between D_i and D_j
R^N	Neighborhood radius of drones
d'_{ik}	Distance between D_i and O_j
kf_i^{δ} and kf_j^{δ}	Computation power of drones i and j
P_{Tx} and P_{Rx}	Transmitting and receiving power
H and t^h	Altitude and time hovering
D^{f} and t^{f}	Distance and time of the flying
D^{P}	Minimum distance between drones for avoiding collision
K	Number of neighbors for each drone
F_i	Fairness of offloading from drone i
$Q(s_t, a_t)$	Q value by s_t and action a_t
$Q_1(s_t, a_t)$	Primary Q network
$Q_2(s_t, a_t)$	Target Q network
s_t, a_t, and r_t	State, action, and reward t
$P_m^{D_i}$	Power consumption of module m in D_i
$T_m^{D_i}$	Computation time of module m in D_i
B_{ij}	Bandwidth between D_i and D_j

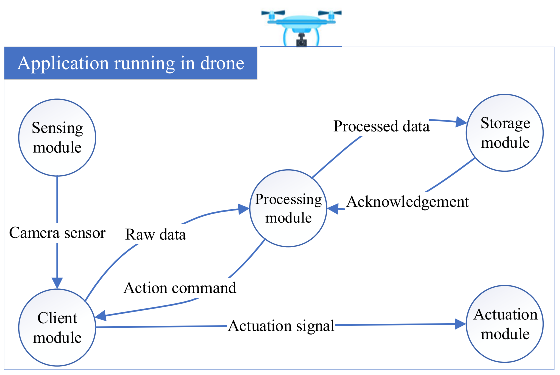

FIGURE 3. Application modules.

processing or object tracking. Therefore, this research does not deal with the content and type of modules, but according to the size of the module and its arrival time in each drone, it examines the execution capacity locally and might make the decision to send it to the neighbors according to their current status. In other words, we look at each module as a black box and decide the best place to run it.

The drones are constantly receiving data from their camera sensors, and the application produces instances of modules according to the data. In fact, each drone deals with a large number of instances of modules during its in-field mission. Each module needs resources to be executed; if the resources in the drone are sufficient, the module is run locally; otherwise, according to the offloading algorithm, it is transferred to another drone. As mentioned in Section I, a module is a task with operations and data.

Assumption 3: Regarding the resource capacity of the drones, they can execute modules locally, but offloading is required when a large number of module processing requests are received on a drone but it is not able to process them. In such a situation, the drone offloads modules to its neighbors. The selection of best neighbor for offloading is processed by FLR strategy. The size of the modules is such that they do not need to be fragmented and a module can be completely moved using 5G communication between drones.

C. LATENCY

Each drone, based on its available computing power, can execute the modules locally or offload them to other neighbor drones. Therefore, the total latency of a module depends on whether it is executed locally or offloaded.

1) COMPUTATION LATENCY FOR LOCAL CASE

In the local execution of drone i (D_i), latency can be expressed as

$$L_i^{\text{Loc}} = \frac{\omega_m}{f_i}, \tag{2}$$

where ω_m is the size of module m, and f_i is the CPU frequency of D_i.

On the other hand, in the offloading case, there are two different latency components, i.e. for communication and computation. There is a first latency component for transferring a module between drones. The second latency component is due to the fact that when the module is placed in a drone D_i for execution, the processors of the drone may be busy and the module will have to wait, thus there is a computation latency.

2) COMPUTATION LATENCY FOR OFFLOADING CASE

The execution of a module on each drone has latency [13]. Here, we offload an entire module, not just a part of it. Thus, the computation latency can be calculated as

$$L_i^{\text{Comp}} = \frac{\omega_m}{f_j}, \tag{3}$$

where ω_m is the size of module m, f_j is the CPU frequency of D_j.

3) COMMUNICATION LATENCY OFFLOADING CASE

The communication latency of that module [13] between D_i and D_j can be given by

$$L_i^{\text{Comm}} = \frac{\beta\omega_m}{R^{\text{DL}}(i,j)}, \tag{4}$$

where S is the size of data for communication and β is a rate of communication size. Since we transmit a module without any division, $\beta\omega_m$ is the module size. Moreover, $R^{\text{DL}}(i,j)$ is the distance-based link rate between D_i and D_j that can be given by

$$R^{\text{DL}}(i,j) = B_{ij}\log_2(1 + \frac{P_{\text{Tx}} d_{ij}^{-\gamma_i} h_i}{N_i}), \tag{5}$$

where B_{ij} is the bandwidth between D_i and D_j, P_{Tx} is transmitting power, h_i is the complex Gaussian channel coefficient which follows the complex normal distribution CN(0,1), N_i is the additive white Gaussian noise with zero mean and variance δ^2, and d_{ij} is the distance between D_i and D_j that can be expressed by

$$d_{ij} = \sqrt{(D_i^x - D_j^x)^2 + (D_i^y - D_j^y)^2 + (D_i^z - D_j^z)^2} \tag{6}$$

γ_i is the path loss exponent [51] that can be given by

$$\gamma_i = \frac{\rho_i}{d_{ij}^2} \tag{7}$$

where ρ_i is the channel power of D_i. Since we are considering a dynamic network, it is required to explain that d_{ij} value is updated after each move of the drones in the environment. This helps us to calculate exactly the communication latency between drones. Moreover, this is also used for calculating the communication energy consumption as will be explained in Subsection IV-D.

4) TOTAL LATENCY

Finally, the total latency can be calculated by identifying the maximum value between local latency as L_i^{Loc} and the sum of the offloading communication latency as L_i^{Comp} and offloading computation latency as L_i^{Comm}, i.e.

$$L_i^{\text{Tot}} = Max(L_i^{\text{Loc}}, [L_i^{\text{Comp}} + L_i^{\text{Comm}}]). \tag{8}$$

D. ENERGY CONSUMPTION

We consider the energy consumption model for the computation, communication, and moving of drones. Regarding the resources capacity of each drone D_i, the modules can execute locally or be transferred to neighbors. Drones consume energy while transferring a module; moreover, the module execution also consumes some energy. Thus, total energy consumption in module offloading is calculated based on the communication, computation, and moving energy consumption.

1) COMPUTATION ENERGY CONSUMPTION

In the local execution in each drone, the computation energy consumption [13] can be expressed as

$$E_i^{\text{Loc}} = \frac{kf_i^{\delta}\omega_m}{f_i} \tag{9}$$

On the other hand, the computation energy consumption in offloading can by given by

$$E_i^{\text{Comp}} = E_i^{\text{Loc}} + \sum_{j=1}^{N} kf_j^{\delta}\frac{S}{f_j} \tag{10}$$

where, kf_i^{δ} and kf_j^{δ} are the computational power of D_i and D_j, respectively and N is the number of modules. Thus, the total computational energy consumption of drones' swarm is the sum of computation energy consumption in the local and offloading modes.

2) COMMUNICATION ENERGY CONSUMPTION

We calculate the communication energy consumption based on the transferring of modules between drones; the total communication energy consumption is calculated as follows.

The transmitting energy consumption [13] can be calculated as

$$E_i^{\text{Tra}} = \sum_{j=1}^{N} P_{\text{Tx}}\frac{\beta S}{R^{\text{DL}}(i,j)} \tag{11}$$

and the receiving energy consumption can be given by

$$E_i^{\text{Rec}} = \sum_{j=1}^{N} P_{\text{Rx}}\frac{\beta \omega_m}{R^{\text{DL}}(i,j)} \tag{12}$$

where P_{Tx} and P_{Rx} are the transmitting and receiving power of D_i and D_j, respectively.

The total communication energy consumption can be given by

$$E_i^{\text{Comm}} = E_i^{\text{Tra}} + E_i^{\text{Rec}}. \tag{13}$$

The total energy consumption is the sum of the communication and computation energy consumption:

$$E_i^{\text{Tot}} = E_i^{\text{Comp}} + E_i^{\text{Comm}}. \tag{14}$$

3) DRONE'S MOVING ENERGY CONSUMPTION

There is a number of parameters in a drone's energy model like time of idle, armed, hovering, and horizontal flying states, take-off speed, the distance of vertical flying upwards and downwards, the altitude of hovering, and payload.

Drones evaluate their neighbors, thus after each rating process according to Eq. 28, the drones with the lowest rating value have to move to another position. The other drones without movement also consume energy in the hovering state.

While moving the drones, we consider a padding distance. If the drones are exactly at the border of the area, then part of their camera coverage will be out of the area, which is useless. The padding distance allows the drones to be positioned at a certain distance from the boundary of the area so that we can use the maximum camera coverage.

In this work, we assume drones are in the air and their energy is calculated as follows [52].

The hovering energy can be expressed as

$$E_i^{\text{H}} = (4.917H + 275.204)t^h, \tag{15}$$

where, H and t^h are the altitude and time of hovering.

The flying horizontally energy can be calculated as

$$E_i^{\text{FH}} = 308.709t^f - 0.852, \tag{16}$$

where, t^f is the flying time.

The flying vertically upward energy can be given by

$$E_i^{\text{FVU}} = 315D^f - 211.261, \tag{17}$$

where, D^f is the distance of this flying. The flying vertically backward can be calculated as

$$E_i^{\text{FVB}} = 68.956D^f - 65.183, \tag{18}$$

Finally, the total moving energy consumption can be expressed as

$$E_i^{\text{Mov}} = E_i^{\text{H}} + E_i^{\text{FH}} + E_i^{\text{FVU}} + E_i^{\text{FVB}}, \tag{19}$$

E. FAIRNESS OF OFFLOADING

The fairness parameter helps us to evaluate the distribution of offloading in all available drones in the neighborhood of each drone.

The swarm of drones has a distributed structure, and our proposed approach for offloading is also distributed. In order to maintain a balance in resource consumption and module processing, we aim at distributing the modules fairly. In fact, the fairness value shows the justice of the offloading strategy in a distributed environment. This parameter, known as Jain index in [53], is based on only the energy consumption of devices. In contrast, we propose to calculate this fairness parameter based on the sum of the total energy consumption and the total latency as:

$$f(E_i^{\mathrm{Tot}}, L_i^{\mathrm{Tot}}) = W_e * E_i^{\mathrm{Tot}} + W_l * L_i^{\mathrm{Tot}}, \tag{20}$$

where, W_e and W_l are weighted coefficients for normalization of the total energy consumption and the total latency. The fairness of D_i can be calculated as

$$F_i = \frac{(\sum_{j=1}^{K} f(E_i^{\mathrm{Tot}}, L_i^{\mathrm{Tot}}))^2}{K * \sum_{j=1}^{K} f(E_i^{\mathrm{Tot}}, L_i^{\mathrm{Tot}})^2} \tag{21}$$

where K is the number of neighbor devices contributed in the offloading, $f(E_i^{\mathrm{Tot}}, L_i^{\mathrm{Tot}})$ is based on the total energy consumption and the total latency. The fairness value is in the range $[1/K, 1]$. Higher values of fairness correspond to higher performance.

V. FEDERATED LEARNING-BASED OFFLOADING IN A SWARM OF DRONES

This section presents the **FL-based offloading with Rating (FLR)** strategy based on cooperation between drones. We divide these steps into four phases, i.e. initialization of drones and application, finding the best learning model in each drone with a rating strategy and offloading, updating learning model by neighbors of drones, and setting the new position of drones. The pseudocode of the proposed approach is given in Algorithm 1. We explain this algorithm based on its four phases as follows.

A. Phase1: INITIALIZATION

All drones are placed in the environment area with random positions. Since we consider heterogeneous drones, their configurations are different. Next, the application modules are mapped onto the drones.

B. Phase2: LOCAL LEARNING AND RATING OF DRONES IN THE SWARM

This phase includes two important parts. 1) The learning runs in a drone locally. 2) On a high-level process, the rating method executes for evaluating drones in a swarm.

1) LOCAL LEARNING BY DQL

All potential neighbors of each drone are evaluated by their distance as per Eq. (6). All D_j with a distance smaller than R^N (Neighborhood radius of drones) are considered as neighbors of D_i. This decision results in some clusters of drones. This helps the drones to perform faster computation offloading together.

Each drone calculates energy consumption, latency, and fairness as per Eqs. (8), (14), and (21), respectively. If a drone can execute the module locally under the given constraints, it does so; otherwise, the offloading process starts.

In each drone, we execute a DQL algorithm. There are two value functions for learning with random assignments [21]. As a result, we have two sets of weights for determining the

Algorithm 1 FLR: Federated Learning With Rating Method

```
1:  Phase 1: Initialization
2:  Place all drones in the area with random positions.
3:  Map all the application modules to drones.
4:  Phase 2: Local learning and rating of drones in the swarm
5:  Setting each drone in a cluster with $d_{ij} < R$.
6:  for It=1 to MaxIteration do
7:      for all drones do
8:          Calculate $L_i^{Tot}$, $E_i^{Tot}$, and $F_i$ as per Eqs.(8), (14), and (21), respectively.
9:          for all Modules do
10:             if module can run locally then
11:                 Run module in current drone.
12:             else
13:                 Run DQL as Algorithm 2.
14:                 Rating current drone by Algorithm 3.
15:                 Phase 3: Cooperative learning
16:                 Update drones' Q with higher rated neighbors as per Eq. (31).
17:                 Sends its Q to neighbors.
18:                 Updates neighbors' Q as per Eq. (31).
19:                 Phase 4: Update drones' position
20:                 Moving the lower rating drones.
21:                 Update neighbors based on new positions.
22:             end if
23:         end for
24:     end for
25: end for
```

greedy policy and its value. In fact, double Q-learning decomposes the max operation in the target into action selection and action evaluation. In DQL, Q is the quality, in this case, it represents how useful a given action is in gaining some future reward. The DQL steps of Algorithm 2 are as follows.

Algorithm 2 DQL

```
    Input: $P_m^{D_i}$ and $T_m^{D_i}$
    Output: Q model
1:  for all Environment steps do
2:      Initialize $Q_1$, $Q_2$, and U by 0.
3:      Observe state $s_t$ and select action $a_t$.
4:      Execute $a_t$, go to state $s_t + 1$, and $r_t$.
5:      U = $(s_t, a_t, r_t, s_t + 1)$.
6:  end for
7:  for all Update steps do
8:      Update $Q(s_t, a_t)$ by Eq. (25).
9:      Run gradient descent as Eq. (26).
10:     Update $Q2(s_t, a_t)$ by Eq. (27).
11: end for
```

First, initialize the primary network Q_1, target network Q_2, and reply buffer U with 0. Then, for all environment steps:

1) Observe state s_t and select action a_t. The system state s_t can be defined by

$$s_t = \{M_t, C_t\} \in S \tag{22}$$

where C_t is the available computation resources for module t (M_t).

For each module, a_t as the action t can be expressed by

$$a_t = \{U_t, D_t\} \tag{23}$$

where U_t is a computation frequency of module t that can be provided by drone D_t.

2) Execute a_t and observe next state s_{t+1} and reward r_t that can be given by

$$R(s_t, a_t) = Avg(P_m^{D_i}, \frac{1}{T_m^{D_i}}) \tag{24}$$

where $R(s_t, a_t)$ is the average value of $P_m^{D_i}$ and $T_m^{D_i}$ as the power and computation time of module m in D_i, respectively.

3) Store (s_t, a_t, r_t, s_{t+1}) in the reply buffer U.

Subsequently, for each update step:

1) $Q(s_t, a_t)$ value can be calculated by

$$Q(s_t, a_t) = r_t + \alpha Q(s_{t+1}, \underset{\alpha'}{\text{argmax}} Q_2(s_{t+1}, \alpha')). \tag{25}$$

where α is the learning rate and α' is the action related to the target network Q_2. Here, an action of global Q can be expressed by *argmax* of α' in Q_2.

2) Execute a gradient descent (GD) step on

$$GD\{(Q(s_t, a_t) - Q_1(s_t, a_t))^2\} \tag{26}$$

3) Update the target network $Q_2(s_t, a_t)$ by

$$Q_2(s_t, a_t) = \alpha Q_1(s_t, a_t) + (1 - \alpha) Q_2(s_t, a_t). \tag{27}$$

2) RATING OF DRONES IN THE SWARM

We propose a rating method for evaluating each drone in a swarm. This helps the FL to find the best drone as a destination for offloading. Each drones can rate all its neighbor drones. The rating value can be given by

$$Rating_j = \frac{w_1 B_j + w_2 E_j + w_3 f_j + w_4 F_j + w_5 P_j}{w_6 L_j} \tag{28}$$

where B_j is the bandwidth of D_j, E_j is the total energy consumption of D_j, f_j and F_j are the CPU frequency and the fairness of D_j, respectively, P_j is the performance of D_j, and L_j is the total latency of D_j. All w_i are weighted coefficients for the normalization of parameters.

The parameter P_j depends on the application type and its mission. In the image object detection application, this parameter can be the accuracy of the detected objects process, which ranges from 0 to 1. Here, we calculate the average of accuracy for object image detection based on proximity of each drone to objects. According to Figure 4, each drone is placed in a random position. Some drones may cover one or more objects, and some other drones may not be able to cover any object due to their poor position.

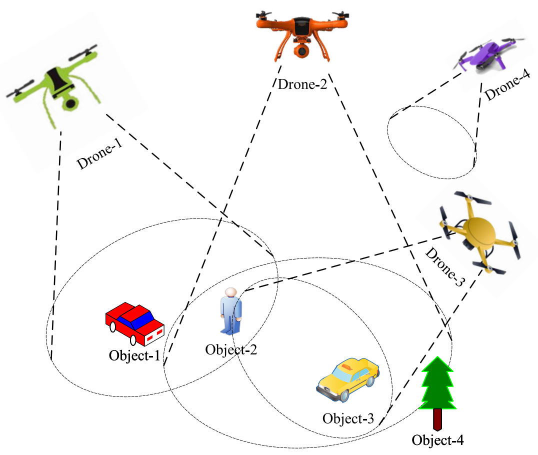

FIGURE 4. Drones with different performances.

For the example application, we assume that the performance of each drone can be expressed by

$$P_j = \begin{cases} 0, & \nexists O_k \in Area^{D_j} \\ 1 - \frac{1}{C}\sum_{k=1}^{C} \frac{1}{d'_{jk}}, & Otherwise \end{cases} \tag{29}$$

where $\nexists O_k \in Area^{D_j}$ means there is no object in the area by D_j. C is the number of tracked objects and d'_{jk} is the distance between drone j as (D_j) and object k as (O_k) that can be given by

$$d'_{jk} = \sqrt{(D_j^x - O_k^x)^2 + (D_j^y - O_k^y)^2 + (D_j^z - O_k^z)^2} \tag{30}$$

According to Figure 4, drone D_1 covers O_1 and O_2, drone D_2 covers O_2, O_3, and O_4, drone D_3 covers O_2 and O_3, and drone D_4 do not cover any objects. However, each drone has a different P_j value as Eq. 29. In practice, P_j may also depend on the application algorithm the configuration of the drones' sensors.

The pseudocode for the proposed rating method is given in Algorithm 3. Each drone sends its configuration and Q model to neighbors. All neighbors, after receiving these data, send their configuration and Q model to their neighbors. Then, each drone, after receiving data from its neighbors, calculates the rating values between neighbors and finds the higher value. After that, the Q model of the drone is updated and the rating and Q model are sent to the neighbors. In the next step, all neighbors receive the configuration and Q model. Then, drones with the lowest rating value move to a new position and send their configuration and Q model to neighbors. Finally, all neighbors update their configuration and Q model.

C. Phase3: COOPERATIVE UPDATING OF THE LEARNING MODEL

As we mentioned in the introduction, the basic traditional FL steps are as follows. 1) It starts by sending a learning model weights from one server to others. 2) Each node sends the local learning model weights to the server. 3) The server aggregates all received learning model weights to create

Algorithm 3 Rating Drones

```
    Input: $D_i$
 1: By $D_i$:
 2: Send configuration and Q to neighbors.
 3: By $D_i$'s neighbors:
 4: Receiving configuration and Q model of drone.
 5: Each neighbor sends its configuration and Q model to
    neighbor.
 6: By $D_i$:
 7: Received configuration and Q model from neighbors.
 8: for j = 1 to $Neighbors_{Last}$ do
 9:     R[j] = Calculate rating value as Eq. (28).
10: end for
11: $[Q_{sorted}, Index_{sorted}]$ = Sort neighbors based on rating
    values.
12: Update Q model by aggregating drones with higher
    $Q_sorted$ values.
13: Send rating and Q model to neighbors.
14: By $D_i$'s neighbors:
15: for j = 1 to $Neighbors_{Last}$ do
16:     Receiving configuration and Q model of $D_j$.
17:     if $R[j]$ is lowest in cluster then
18:         Move $D_j$.
19:         Calculate new neighbors.
20:         Send configuration and Q model to neighbors.
21:     end if
22: end for
23: Update neighbors' configuration and Q model.
```

a global learning model and then sends the global learning model to others.

Our approach is based on this flow, but at the swarm level. In fact, in each cluster of a swarm, each drone aggregates the learning model weights with the received learning model from their neighbors. We execute these steps based on a rating strategy to apply only a higher-rated drone's learning model. In Phase3, each drone updates its Q model based on aggregation of its neighbor drones with higher rating (as we shown in Figure 2). We need to maximize the expected long-term utility of FLR that can be given by the following control policy:

$$W_{t+1} = \sum_{i=1}^{N_r} (\frac{B_t^i}{B_t} * W_{t+1}^i) \tag{31}$$

where W_{t+1} is the new weight, N_r is the number of neighbor drones with higher rating values. B_t^i and B_t are the statistical values of D_i and its neighbors; and B_t is expressed as

$$B_t = \sum_{i \in S} B_t^i \tag{32}$$

where S is a set of drones with higher rating value.

After performing the update, each drone sends its Q model to its neighbors. This process executes in all drones simultaneously. Each drone, as soon as receiving the neighbor Q model, can evaluate it and update its model by aggregating based on neighbors with the higher rating value. After updating the Q model, drone knows the destination of module for offloading.

D. Phase4: UPDATE DRONES' POSITION

To improve the drone's QoS, we move away drones with low performance. This performance is calculated based on their ratings. According to the Pareto principle, 80% of a system's problems occur in 20% of its element [54]. Moreover, due to the high mobility of the drones, they consume a lot of energy, so this percentage is considered.

Thus, we update the position of 20% of the drones with lower ratings and refer to this with η. It should be noted that depending on the rating, different drones may change position each time. After moving the drones, they update their neighbors based on their new position. Finally, with finishing these phases, the process repeats from phase 2 to phase 4 until the maximum simulation time.

E. COMPLEXITY ANALYSIS

Here, we present the computational complexity of the proposed approach, where the number of drones, tracked objects, and modules are N, K, and M, respectively. In Algorithm 1, Lines 1 to 5, we have placed N drones, mapped M modules, and parallel setting of drones in the clusters with complexity $O(N)$, $O(M)$, and $O(N)$, respectively. The complexity until now can be calculated by $O(N + M + N) = O(N + M)$. The complexity of Lines 8 to 11 is $O(1)$. In Line 13, we have the DQL Algorithm 2 with complexity $O(C \cdot N) = O(N)$, where C is the number of environment steps. In Line 14, the rating of drones as Algorithm 3 is parallel-executed with complexity $O(K \cdot N + N) = O(K \cdot N)$. In the following part of Algorithm 1, Lines 15 to 21, we have a complexity of $O(N)$. Finally, the overall complexity of the proposed FLR can be expressed as

$$O(N + M + M \cdot N + M \cdot K \cdot N + M \cdot N) = O(M \cdot K \cdot N) \tag{33}$$

For implementing our sample application as object tracking, we need to use 2 to 20 drones for discovering an area. Also, surveying the recent related works show that other researchers tested their offloading method in a swarm including 10 drones [13], [35], [36] even in a larger environment. The complexity is linear w.r.t the number of nodes N and to the number of tasks/modules, which is also confirmed by the simulation results in Figure 11. Moreover, drones are constantly capturing videos of the environment and have to process them, due to the heterogeneity of the drones, in many cases local computing is not effective and offloading is a better solution.

VI. EVALUATION

In this section, we provide the simulation-based experimental results and evaluate the performance of the proposed offloading strategy. Firstly, we introduce the simulation environment in Subsection VI-A, the evaluation methods in Subsection VI-B and the devices' configuration in Subsection VI-C. Secondly, in Subsections VI-D to VI-I the proposed offloading strategy for drones is analyzed under different metrics and is compared with the benchmark methods.

A. SIMULATION ENVIRONMENT

This work is simulated in the iFogsim [50] simulator as a Java-based library. It is used for resource management projects in IoT, edge, and fog computing. The default architecture of this simulator is hierarchical. This simulator has classes to implement resource management strategies; we have extended some classes i.e. ModulePlacement and ModulePlacementEdgeward for offloading, drone-based application, controller for calculating output metrics. We made those changes to implement different types of nodes involved both in the proposed and the reference distributed computing architectures. The evaluation metrics, e.g. the latency and energy consumption, are implemented according to the equations in Subsections IV-C and IV-D.

B. EVALUATION METHODS

The proposed approach, referred to as FLR, is compared to Random and Greedy [28] offloading approaches, and also compared to an architecture involving edge servers referred to as Hierarchical [14]. These benchmark approaches represent the main state-of-the-art strategies for in-swarm offloading.

- **Random**: This approach is based on an architecture including a swarm of drones with random offloading method. In the underlying algorithm, after each drone has received the Q learning model from its neighbors, the offloading destination for modules are selected randomly without any rating.
- **Greedy**: This approach is based on an architecture including a swarm of drones with greedy offloading method. In the underlying algorithm, after each drone has received the Q learning model from its neighbors, all neighbors are sorted based on their fairness as per Eq. (21). Then the neighbor with maximum fairness value is selected as the offloading destination of the module.
- **Hierarchical**: Here, there are several drones and a number of edge servers and the proposed algorithm (FLR) is now executed in this architecture. For evaluation, we realistically considered the number of edge servers to be equal to 10% of the total number of drones (N'). For example, if there are 50 drones, 5 edge servers are added to the network so that each edge server covers 10 drones.

C. CONFIGURATIONS

The configuration of devices is as shown in Table 3. For the drones, the RAM ranges from 2 to 6 GB, the CPU frequency from 2 to 6 GHz, and the bandwidth from 1 to 3 Mb/s. The edge servers have more powerful resources, i.e. 8 GHz CPU frequency, 12 GB RAM, and 8 Mb/s bandwidth [7].

TABLE 3. System parameters.

Parameter	Value	Parameter	Value
		Drones	
RAM	2 to 6 GB	f_i	2 to 6 GHz
B_i	1 to 3 Mb/s	α, α'	0.2
P_{Tx}	1.258 W	P_{Rx}	1.181 W
d_{ij}	10 to 500 m	d_{ie}	200 to 500 m
R^N	50 m	N^D	10 to 100
k	$1.25 * 10^{-26}$	γ	3
N_i	-100 dBm	h_i	0 or 1
ω_m	1000 KB	η	20%
D^f	10 m	D^p	10 m
ST	1000 s	(x_c, y_c, z_c)	(500 m, 500 m, 500 m)
		Edge servers for Hierarchical	
RAM	12 GB	f_i	8 GHz
B_i	8 Mb/s	N'	10%

The simulated environment is $500\,m * 500\,m * 500\,m$. The minimum distance between drones is 10 m, and the minimum and maximum distances (d_{ie}) between drones and edge servers are 200 and 500 m, respectively. The flying distance D^f (This value is based on the Euclidean distance traveled in three directions x, y, and z.) is 10 m. The transmitting and receiving powers of the drones are 1.258 and 1.181 W [13], respectively. The simulation time (ST) is 1000 s. We assume that each drone has sufficient battery to be active during this time.

D. ANALYSIS OF THE ENERGY CONSUMPTION

Here, we present a comparison of the total energy consumption and also the computation energy consumption of the drones.

Figure 5 shows the comparison results of the total energy consumption for the proposed approach and benchmark approaches introduced above. The horizontal axis in all figures represents the number of drones and the vertical axis represents the energy consumption of the offloading methods. Here the energy consumption is the sum of the individual energy consumption for computation, communication, hovering, and moving of the devices. According to Figure 5, the proposed FLR has a lower energy consumption than Hierarchical, Random, and Greedy. It can be seen that the minimum total energy consumption is $1.58 * 10^9$ J (Joule) for the proposed approach, whereas it is $3.25 * 10^9$, $3.12 * 10^9$, and $3.32 * 10^9$ J for Random, Greedy, and Hierarchical, respectively.

The reason behind this results is that the proposed approach can perform a better offloading between drones. In fact, it can find the best destination for module offloading. A very important point in achieving this amount of improvement in energy consumption is the ranking of drones (based on their ratings). Because of this technique, only some (but not all) low-performance drones have changed their position according to D^f, resulting in reduced energy consumption. The Random and Greedy have higher energy consumption because they do not use the distributed knowledge from their neighbors to find the best drones for offloading. According to Table 3, the edge servers have more RAM, CPU frequency,

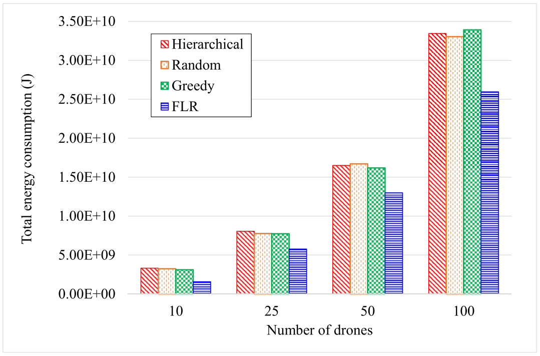

FIGURE 5. Comparison of the total energy consumption of the proposed FLR method against that of the three benchmark methods.

and bandwidth than drones. As a result, the presence of such devices causes higher total energy consumption of the system. Finally, the existence of several relatively powerful edge servers increases the computational and communication energy consumption in the Hierarchical approach.

According to the application model in this research, when it comes to the energy consumption, we focus on the computation energy consumption. Figure 6 shows the average computation energy consumption of all compared methods. The proposed FLR approach has lower energy consumption for communication than the three other methods. This is achieved thanks to its fair and distributed offloading features; indeed, FLR uses a rating method to offload modules to a wide range of drones, not just some drones with sufficient resources. This causes the resources in a swarm of drones to be used in a balanced way.

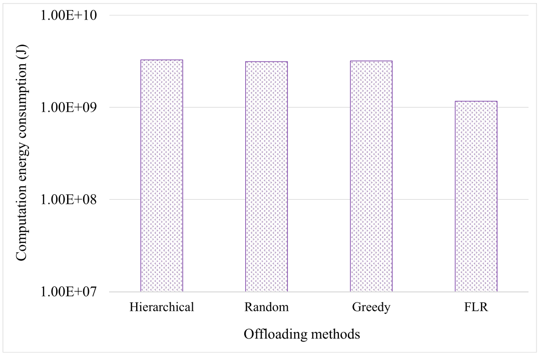

FIGURE 6. Comparison of logarithmic average values of the computation energy consumption of the proposed FLR method against that of the three benchmark methods.

E. ANALYSIS OF THE LATENCY

Figure 7, 8, and 9 show the comparison of the average computation, communication, and total latencies for the proposed approach and the three benchmarks. We consider this metric because it reflects how long it takes for a drone to track an object in the environment, which is a key performance indicator of the application, and especially critical for high-performance real-time applications. The computation,

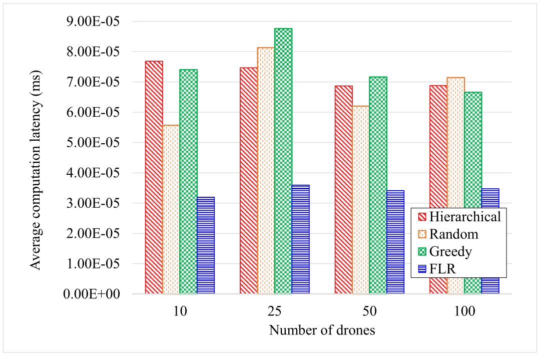

FIGURE 7. Comparison of the computation latency of the proposed FLR method against that of the three benchmark methods.

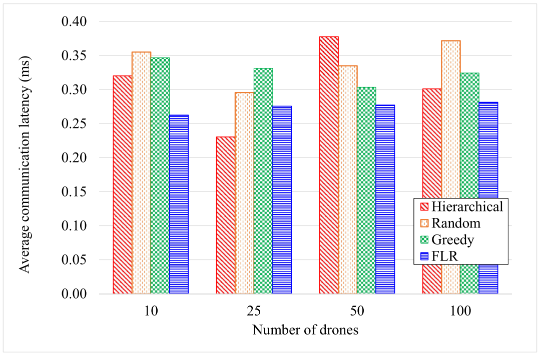

FIGURE 8. Comparison of the communication latency of the proposed FLR method against that of the three benchmark methods.

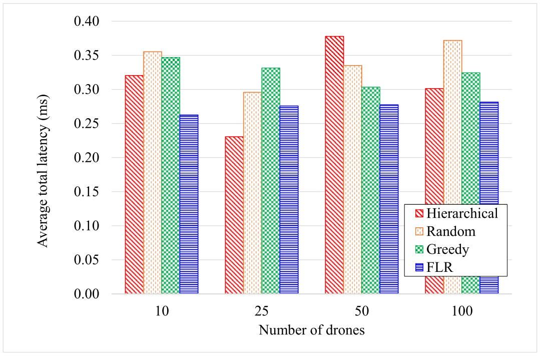

FIGURE 9. Comparison of total latency of the proposed FLR method against that of the three benchmark methods.

communication, and the combined average latencies are presented as follows.

1) AVERAGE COMPUTATION LATENCY

Figure 7 shows the average computation latency of executing modules in drones. This plot illustrates that the proposed approach with the swarm of drones without any centralized devices decreases the average computation latency. The average computation latency for Hierarchical ranges from $6.87 * 10^{-5}$ to $7.68 * 10^{-5}$ ms, Random ranges from $5.57 * 10^{-5}$ to $8.13 * 10^{-5}$ ms, Greedy ranges from $6.66 * 10^{-5}$ to $8.76 * 10^{-5}$ ms, and FLR range from $3.20 * 10^{-5}$ to $3.47 * 10^{-5}$ ms, respectively. The average computation latency of the FLR is clearly lower than that of the other methods. The distribution of modules among a large number of drones and also some edge servers yields fair resource management, so, almost all drones are equally busy and all computations executed with cooperation between drones.

2) AVERAGE COMMUNICATION LATENCY

As we mentioned, the average communication latency refers to the time it takes between sending a module from a drone and receiving that module in another drone. We calculate the average value of this metric. According to Figure 8, the average communication latency for Hierarchical ranges from 0.23 to 0.38 ms, for Random from 0.30 to 0.37 ms, and for Greedy from 0.30 to 0.35 ms. The minimum communication latency provided by the FLR strategy ranges from 0.26 to 0.28 ms. Based on this plot we can say that in most cases the proposed approach can transfer modules with lower latency than that of the other methods.

3) AVERAGE TOTAL LATENCY

According to Figure 9, the proposed approach yields the lowest average total latency. The simulation result shows that the average total latency of Hierarchical ranges from 0.23 to 0.38 ms, Random from 0.30 to 0.37 ms, Greedy from 0.30 to 0.35 ms, and the proposed approach from 0.26 to 0.28 ms. It can also be seen that when the number of drones is 50, the Hierarchical method has the highest latency due to transferring modules between edge servers and drones. However, some edge servers with a significant distance from drones have a larger latency than an architecture including a swarm of drones.

F. ANALYSIS OF THE FAIRNESS

To evaluate the fairness of offloading, let's consider Figure 10. Generally, the minimum fairness is obtained for Random which ranges from 83% to 89% and for Greedy which ranges from 84% to 96%. On the other hand, the maximum value is obtained for the proposed approach which ranges from 92% to 98%. This chart indicates that the proposed approach provides higher distribution of the offloading process. This metric does not have better values in the Hierarchical method than the others due to the forced offloading by edge servers to a number of drones on their domain.

G. ANALYSIS OF THE EXECUTION TIME

Figure 11 indicates the time spent for executing all workloads in the simulator. The proposed approach has lower execution time than the others when the number of drones is 10, 25, and 100. This shows that our strategy based on the fairness of offloading and rating of drones improves the distribution of computations in a swarm. Also, the lower latency of the FLR for decisions about local computing or finding the best destination for offloading decreases the execution time more than the other methods.

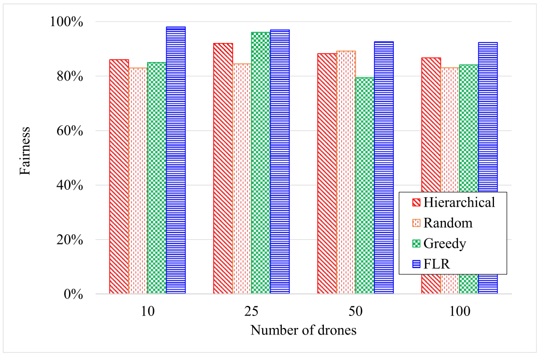

FIGURE 10. Comparison of the fairness in offloading modules of the proposed FLR method against that of the three benchmark methods.

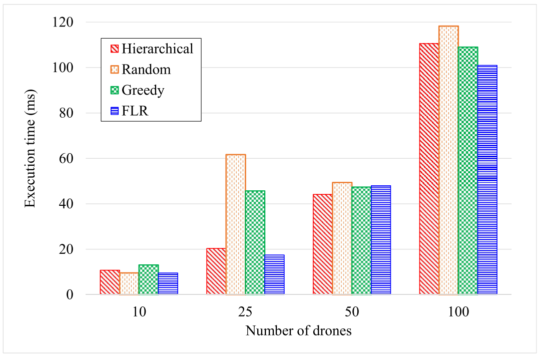

FIGURE 11. Comparison of the execution time of the proposed FLR method against that of the three benchmark methods.

H. ANALYSIS OF THE THROUGHPUT

We also analyze the throughput of the system. Figure 12 shows that the FLR strategy can transfer larger volumes of data at the same time than the other offloading strategies. In the proposed approach, the minimum throughput is $8.53 * 10^3$ MB/s and the maximum is $9.36 * 10^4$ MB/s. The main reason for this result is the distribution of the proposed algorithm and rating of drones.

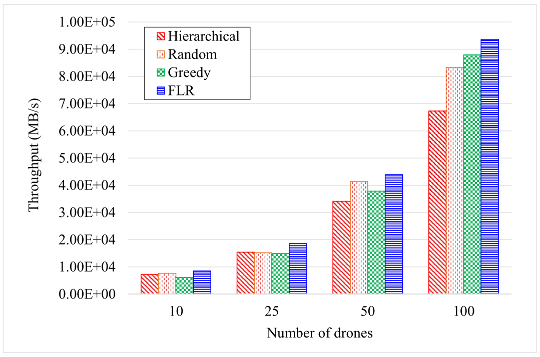

FIGURE 12. Comparison of the throughput of the proposed FLR method against that of the three benchmark methods.

I. IMPROVEMENT PERCENTAGES OF FLR AS COMPARED TO THE BENCHMARK METHODS

Table 4 summarizes the improvement percentages of the proposed FLR approach as compared to the benchmark methods. The rows include total energy consumption, average total latency, throughput, and fairness values. The columns show the name of the parameter, and the difference (in percentage) with the compared methods. The previously presented results and this table show that, as compared to Hierarchical, Random, and Greedy, the proposed FLR approach reduces total energy consumption by −23%, −23%, and −24%, respectively (−23% on average). Regarding total latency, FLR's improvements over Hierarchical, Random, and Greedy are −11, −19, and −16%, respectively (−15% on average). For the throughput metric, FLR's improvements over Hierarchical, Random, and Greedy are +25%, +10%, and +11%, respectively (+18% on average). Finally, in fairness, FLR is superior to Hierarchical, Random, and Greedy by +7%, +11%, and +9%, respectively (+9% on average).

TABLE 4. Improvement percentages of FLR as compared to the other methods.

Parameter	FLR vs. Hierarchical	FLR vs. Random	FLR vs. Greedy	Average
Total energy consumption	-23%	-23%	-24%	-23%
Average total latency	-11%	-19%	-16%	-15%
Throughput	+25%	+10%	+11%	+18%
Fairness	+7%	+11%	+9%	+9%

J. DISCUSSION

In this research, we focused on the computation offloading problem in a swarm of drones and solve it with a distributed strategy. To do this, we considered the limitations and assumptions as follows.

1) We assumed all drones are in the air considering a suitable distance for avoiding a collision. This can be extended as a challenge for more efficiency.

2) The camera's angle of all drones is fixed, thus if an object is in the camera coverage of a drone then the drone can track the object. Optimizing the camera's angle can save in the moving of each drone in a swarm. This can be another extension of this research.

3) Since the focus of this research was not image or video processing optimization, regarding the quality of object tracking by drones, we only considered the distance between drones and tracked objects for the performance. This aspect could benefit from further research.

4) We assume that each drone has sufficient battery to be active during the simulation time. Working on renewable energy resources can be considered as another extension of this work.

5) Drone's moving energy consumption was modeled and calculated according to research that we cited in the same subsection. Implementing a swarm of drones with the proposed approach in this research would benefit from the exact analysis and designing the system model based on the devices.

VII. CONCLUSION

This paper proposes a novel federated learning based fast and fair offloading strategy with a rating method in a swarm of drones. The rating is constantly computed for each drone (on offloading events) based on a set of its current properties such as available performance, energy, communication, and fairness that reflect a drone's potential to execute the received task.

The experimental comparison with state-of-the-art benchmark methods show that, on average, the proposed distributed offloading strategy can reduce energy consumption and latency by 23% and 15%, as well as increase throughput and fairness by 18% and 9%, respectively. Thus, a swarm of drones in MEC can be used for different missions with improved resource management in a distributed structure.

Still, there are a few limitations to this work, highlighted in what follows. For presenting a fully complete system model for a swarm, we would need to consider a high performance trajectory planning for placing and moving the drones. This challenge can be solved by different strategies such as machine learning. For future work, trajectory planning of drones can be investigated, i.e. the drones can predict their trajectories based on energy consumption, latency, and QoS. FL can provide the basis for trajectory planning based on learning of local energy consumption and collaboration between devices. Using or proposing a suitable channel modeling for a swarm of drones is a second aspect to be considered. Lastly, due to the already complex issue of resource management, the simultaneous design of resource management and application has not be conducted in this paper; this is something worth of further research.

REFERENCES

[1] S. Zaidi, M. Atiquzzaman, and C. T. Calafate, "Internet of flying things (IoFT): A survey," *Comput. Commun.*, vol. 165, pp. 53–74, Jan. 2021, doi: 10.1016/j.comcom.2020.10.023.

[2] W. Feng, J. Tang, Y. Yu, J. Song, N. Zhao, G. Chen, K.-K. Wong, and J. Chambers, "UAV-enabled SWIPT in IoT networks for emergency communications," *IEEE Wireless Commun.*, vol. 27, no. 5, pp. 140–147, Oct. 2020, doi: 10.1109/MWC.001.1900656.

[3] J. Zheng, T. Yang, H. Liu, T. Su, and L. Wan, "Accurate detection and localization of UAV swarms-enabled MEC system," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5059–5067, Jul. 2021, doi: 10.1109/TII.2020.3015730.

[4] K. Kuru, "Planning the future of smart cities with swarms of fully autonomous unmanned aerial vehicles using a novel framework," *IEEE Access*, vol. 9, pp. 6571–6595, Jan. 2021, doi: 10.1109/ACCESS.2020.3049094.

[5] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: The QoS perspective," *IEEE Netw.*, vol. 33, no. 2, pp. 36–43, Mar. 2019, doi: 10.1109/MNET.2019.1800222.

[6] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019, doi: 10.1016/j.sysarc.2019.02.009.

[7] B. Liu, W. Zhang, W. Chen, H. Huang, and S. Guo, "Online computation offloading and traffic routing for UAV swarms in edge-cloud computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8777–8791, Aug. 2020, doi: 10.1109/TVT.2020.2994541.

[8] Q. Zhang, J. Chen, L. Ji, Z. Feng, Z. Han, and Z. Chen, "Response delay optimization in mobile edge computing enabled UAV swarm," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3280–3295, Mar. 2020, doi: 10.1109/TVT.2020.2964821.

[9] H. Xu, W. Huang, Y. Zhou, D. Yang, M. Li, and Z. Han, "Edge computing resource allocation for unmanned aerial vehicle assisted mobile network with blockchain applications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 5, pp. 3107–3121, May 2021, doi: 10.1109/TWC.2020.3047496.

[10] L. Sun, L. Wan, and X. Wang, "Learning-based resource allocation strategy for industrial IoT in UAV-enabled MEC systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5031–5040, Jul. 2021, doi: 10.1109/TII.2020.3024170.

[11] A. Mukherjee, S. Misra, V. S. P. Chandra, and N. S. Raghuwanshi, "ECoR: Energy-aware collaborative routing for task offload in sustainable UAV swarms," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 4, pp. 514–525, Oct. 2020, doi: 10.1109/TSUSC.2020.2976453.

[12] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, 2nd Quart., 2021, doi: 10.1109/COMST.2021.3058573.

[13] X. Hou, Z. Ren, J. Wang, S. Zheng, W. Cheng, and H. Zhang, "Distributed fog computing for latency and reliability guaranteed swarm of drones," *IEEE Access*, vol. 8, pp. 7117–7130, Jan. 2020, doi: 10.1109/ACCESS.2020.2964073.

[14] H. Zhang and L. Hanzo, "Federated learning assisted multi-UAV networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 14104–14109, Nov. 2020, doi: 10.1109/TVT.2020.3028011.

[15] U. Challita, A. Ferdowsi, M. Chen, and W. Saad, "Machine learning for wireless connectivity and security of cellular-connected UAVs," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 28–35, Feb. 2019, doi: 10.1109/MWC.2018.1800155.

[16] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020, doi: 10.1109/COMST.2020.2986024.

[17] X. Lu, L. Xiao, C. Dai, and H. Dai, "UAV-aided cellular communications with deep reinforcement learning against jamming," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 48–53, Aug. 2020, doi: 10.1109/MWC.001.1900207.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, Nov. 2018.

[19] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," Nov. 2018, pp. 219–354, vol. 11, no. 3, *arXiv:1811.12560*. [Online]. Available: http://arxiv.org/abs/1811.12560

[20] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[21] H. A. A. Van Guez and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2016, vol. 30, no. 1, pp. 2094–2100.

[22] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019, doi: 10.1109/MNET.2019.1800286.

[23] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53841–53849, Mar. 2020, doi: 10.1109/ACCESS.2020.2981430.

[24] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020, doi: 10.1109/MCOM.001.1900461.

[25] Y. Wang, Z. Su, N. Zhang, and A. Benslimane, "Learning in the air: Secure federated learning for UAV-assisted crowdsensing," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1055–1069, Apr. 2021, doi: 10.1109/TNSE.2020.3014385.

[26] Z. Ullah, F. Al-Turjman, U. Moatasim, L. Mostarda, and R. Gagliardi, "UAVs joint optimization problems and machine learning to improve the 5G and beyond communication," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107478, doi: 10.1016/j.comnet.2020.107478.

[27] S. Yu, X. Gong, Q. Shi, X. Wang, and X. Chen, "EC-SAGINs: Edge computing-enhanced space-air-ground integrated networks for internet of vehicles," *IEEE Internet Things J.*, early access, Jan. 19, 2021.

[28] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3984–3997, Sep. 2019, doi: 10.1109/TCYB.2019.2935466.

[29] H. Wang, H. Ke, and W. Sun, "Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 180784–180798, Oct. 2020, doi: 10.1109/ACCESS.2020.3028553.

[30] B. Yang, X. Cao, C. Yuen, and L. Qian, "Offloading optimization in edge computing for deep-learning-enabled target tracking by internet of UAVs," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9878–9893, Jun. 2021, doi: 10.1109/JIOT.2020.3016694.

[31] R. Li, X. Li, J. Xu, F. Jiang, Z. Jia, D. Shao, L. Pan, and X. Liu, "Energy-aware decision-making for dynamic task migration in MEC-based unmanned aerial vehicle delivery system," *Concurrency Comput., Pract. Exper.*, pp. 1–18, Nov. 2020, Art. no. e6092, doi: 10.1002/cpe.6092.

[32] X. Zhang, J. Zhang, J. Xiong, L. Zhou, and J. Wei, "Energy-efficient multi-UAV-enabled multiaccess edge computing incorporating NOMA," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5613–5627, Jun. 2020, doi: 10.1109/JIOT.2020.2980035.

[33] J. Zhang, L. Zhou, F. Zhou, B.-C. Seet, H. Zhang, Z. Cai, and J. Wei, "Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2114–2125, Feb. 2020, doi: 10.1109/TVT.2019.2960103.

[34] A. Asheralieva and D. Niyato, "Distributed dynamic resource management and pricing in the IoT systems with blockchain-as-a-service and UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1974–1993, Mar. 2020, doi: 10.1109/JIOT.2019.2961958.

[35] W. Liu, Y. Xu, N. Qi, K. Yao, Y. Zhang, and W. He, "Joint computation offloading and resource allocation in UAV swarms with multi-access edge computing," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2020, pp. 280–285, doi: 10.1109/WCSP49889.2020.9299713.

[36] K. Yao, J. Chen, Y. Zhang, L. Cui, Y. Yang, and Y. Xu, "Joint computation offloading and variable-width channel access optimization in UAV swarms," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6, doi: 10.1109/GLOBECOM42002.2020.9322587.

[37] K. Yao, Y. Xu, J. Chen, Y. Gong, Y. Yang, C. Yao, and Z. Du, "Distributed joint optimization of deployment, computation offloading and resource allocation in coalition-based UAV swarms," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2020, pp. 207–212, doi: 10.1109/WCSP49889.2020.9299672.

[38] R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Netw.*, vol. 35, no. 1, pp. 148–155, Jan. 2021, doi: 10.1109/MNET.011.2000295.

[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[40] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7, doi: 10.1109/ICC.2019.8761315.

[41] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," Dec. 2017, *arXiv:1712.01887*. [Online]. Available: http://arxiv.org/abs/1712.01887

[42] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," 2019, *arXiv:1905.06641*. [Online]. Available: http://arxiv.org/abs/1905.06641

[43] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018, *arXiv:1812.07210*. [Online]. Available: http://arxiv.org/abs/1812.07210

[44] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," in *Proc. NIPS Workshop Federated Learn. Data Privacy Confidentiality*, Dec. 2019, pp. 1–8.

[45] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, Sep. 2018, pp. 21–28.

[46] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*. [Online]. Available: http://arxiv.org/abs/1902.01046

[47] J. S. Ng, W. Y. B. Lim, H.-N. Dai, Z. Xiong, J. Huang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Joint auction-coalition formation framework for communication-efficient federated learning in UAV-enabled internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2326–2344, Apr. 2021, doi: 10.1109/TITS.2020.3041345.

[48] W. Y. B. Lim, J. Huang, Z. Xiong, J. Kang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Towards federated learning in UAV-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5140–5154, Aug. 2021, doi: 10.1109/TITS.2021.3056341.

[49] S. Tang, W. Zhou, L. Chen, L. Lai, J. Xia, and L. Fan, "Battery-constrained federated edge learning in UAV-enabled IoT for B5G/6G networks," *Phys. Commun.*, vol. 47, Aug. 2021, Art. no. 101381, doi: 10.1016/j.phycom.2021.101381.

[50] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, Jun. 2017, doi: 10.1002/spe.2509.

[51] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018, doi: 10.1109/TWC.2017.2789293.

[52] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance," *IEEE Access*, vol. 6, pp. 58383–58394, Oct. 2018, doi: 10.1109/ACCESS.2018.2875040.

[53] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," *Comput. Elect. Eng.*, vol. 44, pp. 218–240, May 2015, doi: 10.1016/j.compeleceng.2015.02.006.

[54] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Highly reliable architecture using the 80/20 rule in cloud computing datacenters," *Future Gener. Comput. Syst.*, vol. 77, pp. 77–86, Dec. 2017, doi: 10.1016/j.future.2017.06.011.

DADMEHR RAHBARI received the B.Sc. degree in computer engineering from Iran University of Science and Technology, Iran, in 2007, the M.Sc. degree in artificial intelligence from IAUM, Iran, in 2010, and the Ph.D. degree in information security from the Department of Computer Engineering and Information Technology, University of Qom, Iran, in 2020. From 2008 to 2020, he was a designer, a programmer, and a consultant in software engineering and information technology with companies, and a Lecturer and a Researcher with UAST, IAU, TVU, PNU, and MSRT universities in Iran. He is currently a Postdoctoral Researcher with the Communication System Research Group, Thomas Johann Seebeck Department of Electronics, School of Information Technologies, Tallinn University of Technology, Estonia. His major research interests include resource management and security in distributed computing and machine learning. He has been invited to be a program committee member and an editorial board member of some international conferences and journals.

MUHAMMAD MAHTAB ALAM (Senior Member, IEEE) received the M.Sc. degree in electrical engineering from Aalborg University, Denmark, in 2007, and the Ph.D. degree in signal processing and telecommunication from the INRIA Research Center, University of Rennes1 France, in 2013. He joined Swedish College of Engineering and Technology, Pakistan, in 2013, as an Assistant Professor. He did his postdoctoral research at Qatar Mobility Innovation Center, Qatar, from 2014 to 2016. In 2016, he joined as the European Research Area Chair and an Associate Professor with Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology, where later he was elected as a Professor, in 2018. Since 2019, he has been the communication systems research group leader. He has more than 15 years of combined academic and industrial multinational experiences while working in Denmark, Belgium, France, Qatar, and Estonia. He has several leading roles as PI in multimillion Euros international projects funded by the European Commission (H2020-ICT-2019-3), "951867," and NATO-SPS (G5482), Estonian Research Council (PRG424), and Telia Industrial Grant. He is the author or coauthor of more than 100 research publications. He is also a Contributor in two standardization bodies (ETSI SmartBAN and IEEE-GeeenICT-EECH), including "Rapporteur" of work item: DTR/SmartBAN-0014 and Applying SmartBAN MAC (TS 103 325). His research interests include wireless communications—connectivity, NB-IoT 5G/B5G services and applications, and low-power wearable networks for SmartHealth.

YANNICK LE MOULLEC (Senior Member, IEEE) received the M.Sc. degree from the Université de Rennes I, France, in 1999, and the Ph.D. and HDR (accreditation to supervise research) degrees from the Université de Bretagne Sud, France, in 2003 and 2016, respectively. From 2003 to 2013, he successively held a Postdoctoral Researcher position, an Assistant Professor position, and an Associate Professor position with the Department of Electronic Systems, Aalborg University, Denmark. He then joined the Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology, Estonia, first as a Senior Researcher, from 2013 to 2016, and has been on a professorship, since 2017. He has supervised or co-supervised more than 50 M.Sc. students and 11 Ph.D. students. He has been involved in more than 20 projects, including five as a PI, a co-PI, or a co-main applicant; one such notable project was the H2020 COEL ERA-Chair project, from 2015 to 2019. His research interests include embedded systems, reconfigurable systems, the IoT, and the application thereof. He is a member of the IEEE Sustainable ICT Technical Community and the IEEE Circuits and Systems Society.

MAKSIM JENIHHIN (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from Tallinn University of Technology, in 2004 and 2008, respectively. He is currently a Professor of computing systems reliability with the Department of Computer Systems, Tallinn University of Technology. He has published more than 120 research papers on these topics and served in executive committees for a number of IEEE conferences, including the Program Chair role for ETS, DDECS, and NORCAS; a guest editor for journals; and a PC member for many scientific events. He participated in numerous EU research projects in FP6, FP7, H2020, and COST frameworks. He is also a Project Coordinator for H2020 MSCA ITN RESCUE (2017–2021)—Interdependent Challenges of Reliability, Security, and Quality in nanoelectronic systems design, and a Founder of the Biannual European–Latin American Summer School on Design Test and Reliability. His research interests include nanoelectronics lifetime reliability and manufacturing test topics, deep learning methods and HW architectures, electronic design automation (EDA) tools and methodologies for computing systems modeling, verification and debug, and interference analysis of functional and extra-functional design aspects, such as resilience and security.