UAV-enabled Mobile Edge Computing for Resource Allocation using Cooperative Evolutionary Computation

Shidrokh Goudarzi, *Member, IEEE,* Seyed Ahmad Soleymani,Wenwu Wang, *Senior Member, IEEE,* Pei Xiao, *Senior Member, IEEE*

Abstract—Edge computing is a viable paradigm for supporting the Industrial Internet of Things deployment by shifting computationally demanding tasks from resource-constrained devices to powerful edge servers. In this study, mobile edge computing (MEC) services are provided for multiple ground mobile nodes (MNs) through a time-division multiple access protocol using the unmanned aerial vehicle (UAV)-enabled edge servers. Remotely controlled UAVs can serve as MEC servers due to their adaptability and flexibility. However, the current MEC approaches have proven ineffective in situations where the number of MNs rapidly increases, or network resources are sparsely distributed. Furthermore, suitable accessibility across wireless networks via MNs with an acceptable quality of service is a fundamental problem for conventional UAV-assisted communications. To tackle this issue, we present an optimized computation resource allocation model using cooperative evolutionary computation to solve the joint optimization problem of queue-based computation offloading and adaptive computing resource allocation. The developed method ensures the task computation delay of all MNs within a time block, optimizes the sum of MN's accessibility rates, and reduces the energy consumption of the UAV and MNs while meeting task computation restrictions. Moreover, we propose a multilayer data flow processing system to make full use of the computational capability across the system. The top layer of the system contains the cloud centre, the middle layer contains the UAV-assisted MEC (U-MEC) servers, and the bottom layer contains the mobile devices. Our numerical analysis and simulation results prove that the proposed scheme outperforms conventional techniques such as equal offloading time allocation and straight-line flight.

Index Terms—Mobile edge computing (MEC), queue-based computation offloading, resource allocation, unmanned aerial vehicle (UAV)

1 Introduction

The growing use of mobile nodes (MNs) is accelerating the development of the Internet of Things (IoT) and the implementation of current mobile applications that demand sophisticated capabilities, such as autonomous navigation and unmanned driving. Because of the inadequate computing capabilities and battery capacity, the quality of experience of compute-intensive services operated on MNs, as well as the MN lifetime, is adversely affected [1]. The mobile topology has intermittent communication links for high-speed data transfer; in addition, the requirements of services with varied quality of service (QoS) conditions change dynamically.

Mobile edge computing (MEC) has been recognised as a potential technique for reaping the benefits of heterogeneous IoT applications, as it can utilize diverse cloud resources such as storage and computation capabilities closer to the MNs [2]. MEC is a potential concept that places cloud servers near the MNs of mobile networks [3]. Offloading compute processes to the MEC server can improve both the quality of computing and the battery life. However, in some cases with inadequate infrastructure, such as disaster response, military mobility, emergency aid, or in remote locales such as forests, mountains, and wetlands, the technology is infeasible. Therefore, unmanned aerial vehicle (UAV)-enabled MEC was envisioned and developed as a viable option to address this drawback [4]. The previous studies [4], [5] have mainly focused on the computation and communication offloading of MNs, and the computation time at the UAVs has not been considered. Nevertheless, the computation time duration at the UAVs cannot be neglected in a real situation. Moreover, the existing studies [6], [7] focused on partial or binary computational tasks. However, task streaming over a fixed period of time has received little attention in UAV-enabled MEC.

This study is motivated by utilizing UAVs as MEC servers in wireless networking. Against this background, this study proposes a joint computation offloading and adaptive computing resource allocation system, where the UAVs serve as edge servers to provide edge computing services to the MNs. The objective of this study is to minimise the energy consumption of the system used by all MNs while guaranteeing effective task computation for all MNs within a time block, by jointly addressing the queue-based

S. Goudarzi was with the Centre for Vision Speech and Signal Processing (CVSSP), University of Surrey, Guildford GU2 7XH, U.K. She is now with the School of Computing and Engineering, University of West London, St Mary's Rd, Ealing, London, W5 5RF, UK. (e-mail: shidrokh.goudarzi@uwl.ac.uk).

S. A. Soleymani is with the Centre for Vision Speech and Signal Processing (CVSSP), University of Surrey, Guildford GU2 7XH, U.K. (e-mail: s.soleymani@surrey.ac.uk).

W. Wang is with the Centre for Vision Speech and Signal Processing (CVSSP), University of Surrey, Guildford GU2 7XH, U.K. (e-mail: w.wang@surrey.ac.uk).

P. Xiao is with the 5G&6G Innovation Centre, Institute for Communication Systems (ICS), University of Surrey, Guildford, U.K. (e-mail: p.xiao@surrey.ac.uk).

offloading and resource allocation.

This is a non-convex nonlinear optimization problem, which is an NP-hard problem. Cooperative evolution [8] has inherent parallelism due to the use of the divide-and-conquer technique, making it applicable to a distributed network. Moreover, the distributed cooperative evolutionary method has a high level of scalability, allowing it to address problems with high dimensions and computational demands. By suggesting novel decomposition methods, some studies [9]–[12] changed the grouping structure dynamically throughout the optimization process. However, most of them still employ the classical cooperative evolutionary design which is an even round-robin method that considers all subcomponents equally, where the subcomponent refers to the UAV processors used for task computation.

Different from this, we present a cooperative evolutionary computation approach using software-defined networking (SDN) in a multilayer data flow processing system where the computation resources are optimized during the entire process. More specifically, we develop an optimized computing resource allocation (OCRA) model for assessing the viability of the subcomponents and allocating resources accordingly in the SDN control layer, where SDN is used to control the network connectivity across the data centre with dynamic programming. The top layer is the cloud centre (CC), the middle layer contains the UAV-assisted MEC (U-MEC) servers, and the bottom layer contains the mobile devices, with centralised control of the network elements [2]. To minimise congestion and reduce latency, the SDN controller divides and assigns the network flows into various routes [13]. In cloud connectivity, SDN refers to a control model in which a central controller makes decisions on resource distribution [14].

To effectively allocate the computational resources to the subcomponents, the OCRA model utilizes various strategies: (i) evaluate the degree of contributions for each subcomponent; (ii) calculate the priority of the subcomponents and distribute the resources accordingly; (iii) evaluate the cost function based on the computation time for selecting the best UAV for serving the MN's population; (iv) utilize a verification plan inside a shared pool layer to assist the optimizer in making effective use of a given computational resource and record the role of each subcomponent to achieve the global objective; and (v) utilize the queue based offloading algorithm to make optimal offloading decisions for choosing the subcomponents. The following are the key contributions of this paper:

1) We formulate an optimization problem for computation offloading and resource allocation in wireless networks with UAV-enabled edge servers.
2) We propose a two-layer cooperative co-evolution model to address the energy consumption and computation time minimization issues. The computing task is assigned to multiple UAV-enabled edge servers based on the dimension of the MN population and the priority constraints among the UAV processors.
3) We apply techniques to distribute the resources adaptively, with the priority ranking of computing tasks of the MNs in the first layer, and the distribution of MNs tasks across different UAV processors by a population-distribution model in the second layer.
4) We propose a queue-based offloading algorithm to make optimal offloading decisions with UAVs to facilitate the allocation of MNs' queued computing tasks and minimize the task execution latency.

The remainder of the paper is organised as follows. Section 2 presents related research on computation offloading in UAV-enabled MEC networks. Section 3 presents our proposed model for the problem of joint optimization for the resource allocation and computation offloading control. Section 4 describes the proposed algorithm in detail. Section 5 presents a comparison of the proposed approach against state-of-the-art methods. Finally, concluding remarks are drawn in Section 6.

2 Related work

UAVs have attracted considerable attention from both academia and industry due to their high flexibility in deployment. UAVs have been used in a variety of wireless communication applications, including non-orthogonal multiple access networks [15], mmWave communications [16], and caching [17]. A previous study presented the three-dimensional (3D) coverage performance of the cellular network-connected UAVs that function as aerial nodes [18]. UAV relaying was also reported as an important application that may effectively enhance the coverage for communications [19]. The UAV-enabled MEC features dependency on line-of-sight (LoS) connectivity and controlled mobility management. For example, in a previous study [5], the researchers created a dispersed deployment strategy for UAVs, which maximised the average distance between the UAV and MN. Nevertheless, they assumed that the UAVs served all MNs with the same data rate instead of introducing various data rates for various MNs [5].

Numerous UAV deployment methods have been designed to address the requirements of MNs with varying data rates [20]. However, employing evolutionary-based solutions for the combined resource allocation and deployment problem has not been addressed in the literature. We explored stochastic task models in the MEC scenario with several servers and MNs to reduce the computation for the overall long-term weighted average of power in the system [21]. Although those problems have been addressed in terrestrial cellular environments, MEC server mobility management has received little attention. Most previous studies related to UAV deployment in communication systems have envisaged their main applications as moving relays [22] or flying base stations [23]. Combined communication and computing resource allocation have been studied in the scenario of multi-user with single server MEC [24]–[27]. However, they did not take offload computation into account.

Our work differs from the previous studies [22], [24]–[27] in the following aspects. First, our study analyzes the situations in which the UAV-enabled MEC server can cater for multiple MNs concurrently, whereas previous studies only considered the scenario in which the MEC server can sequentially serve multiple MNs. Second, this work focuses on the cooperative optimization between UAV task

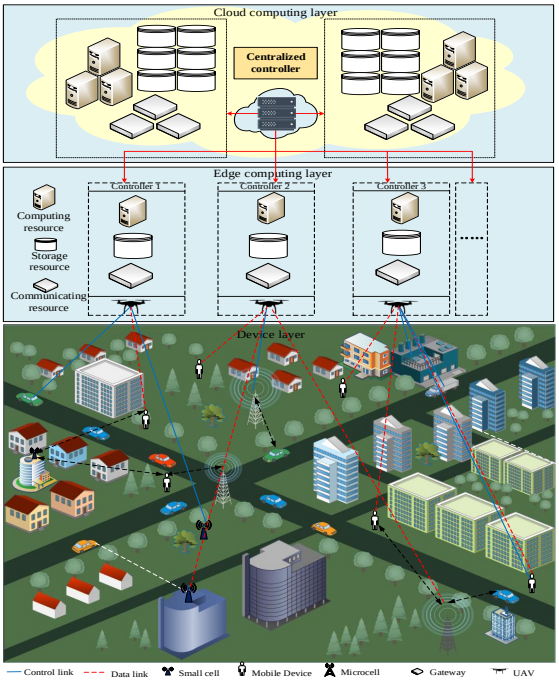

Figure 1: Network model.

execution and MN task execution, whereas previous studies have focused only on one of these approaches. Third, for the first time, the priority among different UAV processors in computing resource allocation is considered in our method. In addition, we present both locally and globally optimal solutions to the joint optimization issue, whereas the previous studies have only reported locally optimized solutions.

3 Network Model and Problem Formulation

We propose a UAV-enabled MEC system, as depicted in Figure 1, whereby each UAV performs MEC server computation for the overlaying MNs. There are two primary modules in the SDN controller. The first module is used to keep track of the tasks. This module stores all the MN mission data. It also determines which tasks are computed locally or offloaded to the edge for processing. The other module is the edge server module. Its data indicate how much memory and CPU are available on the server, as well as the loading of the server. The objective is to fly the UAV from an initial location to a designated destination and offload the MN computing tasks to the MEC server for implementation. The UAV-enabled MEC servers, represented by a set $\mathcal{J} = \{1, ..., j, ..., J\}$, are evenly distributed in the area to serve the MNs on the ground in the present study. Furthermore, $\mathcal{I} \triangleq \{1, ..., i, ..., I\}$ and $\mathcal{N} \triangleq \{1, ..., n, ..., N\}$ are used to represent the sets of the MNs and time slots, respectively. The set $\mathcal{T} = \{1, ..., t, ..., T\}$ is split into N time slots and refers to the time taken to complete a task, with a slot length of τ, $T = \tau N$.

The UAV provides equal bandwidth allocation to all MNs by using frequency division multiple access (FDMA). We assume that the UAV flies at a constant altitude $H > 0$ above the ground. The positions of the MNs and UAVs as edge servers are represented using the 3D Cartesian coordinate system in this article. The 3D coordinates of the UAV j are denoted as $\mathbf{u}_j$=(X_j, Y_j, H). The 3D coordinates of the MN i are denoted as $\mathbf{m}_i$=$(x_i, y_i, 0)$. At the UAV, the 3D coordinates of all MNs are recognised in advance for resource allocation and position design. The line-of-sight (LoS) links should be considered between all MNs on the ground and the UAV because the UAV flies relatively high and the probability of the UAV dispersing is minimal [28]. The effects of the LoS paths are dominant in comparison with the non-line-of-sight (NLoS) pathways [22]. Thus, the channel power gain from MN i to the UAV j is modelled according to a previous study [28],

$$h_{ji} = \frac{g_0}{\|\mathbf{u}_j - \mathbf{m}_i\|^2}, i \in \mathcal{I} \tag{1}$$

where g_0 represents the reference channel gain at 1 meter away from MN i, and $\|\cdot\|$ represents an l_2 norm. For MN i, the computation task with $A_i(t) > 0$ bits at time slot t is separated into two sections with $u_i \geq 0$ and $v_i \geq 0$ bits. The latter is computed locally, while the former is sent to the MEC server with UAV capabilities.

Interference between the MNs and MEC servers can be avoided during the data transmission phase if the orthogonal FDMA (OFDMA) technology is employed for data transmissions between the MEC servers and the MNs. The matrix $\mathbf{E}$ describes the channel selection by the MNs, with each element $E_{i,t}$ representing the number of channels filled by MN i at time t. We also constructed a new matrix $\mathbf{F}$ to represent the resource assignment, in which $F_{i,j}$ ($i \in \mathcal{I}$ and $j \in \mathcal{J}$) are the computing resources allocated to MN i from MEC server j. For call admission control, we used a QoS-based approach that MN is assigned a rate requirement R_i. When a new call arrives, the call admission policy examines the bandwidth; if the bandwidth is available, the call is connected; otherwise, the call is terminated. The transmission bandwidth of UAV j for each MN is defined as $\tilde{B}_j$ such that $\tilde{B}_j = \frac{B_j}{|\mathcal{I}_j|}$, where B_j is the available bandwidth of UAV j, $\mathcal{I}_j \subset \mathcal{I}$ and $|\mathcal{I}_j|$ is the number of MNs served by UAV j. Each UAV j shares a proportion of the resource, defined as α_j, with the macro MNs, and $(1 - \alpha_j)$ is allocated to other MNs of the UAV. Each MN's bandwidth demand must be equal or less than allocated bandwidth, which is denoted by $\frac{(1-\alpha_j)\cdot B_j}{|\mathcal{I}_j|} \leq max\tilde{B}_j$. In order to guarantee QoS, the criterion for admission control [19] is given as follows:

$$\frac{max\tilde{B}_j \cdot |\mathcal{I}_j|}{B_j} \leq 1 \tag{2}$$

Furthermore, the UAV may serve $|\mathcal{I}_j|$ ground MNs simultaneously owing to implementation of the FDMA mechanism. The flying UAVs act as edge servers to deliver edge computing services to MNs because MNs have extremely limited processing capabilities and energy, as depicted in Figure 1. In this article, we consider UAVs which have efficient processors for performing computationally demanding tasks as those in [29], [30]. In a mobile topology, the communication lines demand high-speed data transfer in addition to the continuously changing needs of services with varying QoS requirements. Our goal is to optimize the sum of the average rates of the MNs while maintaining the

QoS of the MNs by optimizing the computation offloading and resource allocation control jointly. This is a non-convex mixed integer problem that can be divided into sub-components, which can be optimized by the cooperative evolutionary computation technique. The signal-to-noise ratio (SINR) [29], represented by γ_i, of any MN i is expressed as

$$\gamma_i = pg_0/\sigma^2 \tag{3}$$

where p is the transmission power of the UAV, and σ^2 is the noise power. Based on the Shannon capacity formula, the achievable data rate of MN i, represented by r_i, from the UAV j can be expressed as $r_i = Wlog_2(1+\gamma_i)$ and W refers to the bandwidth allocated to MN i. We defined a variable $a_{i,j}$ to indicate the connectivity of MN i to UAV j. If the achievable data rate r_i of MN i is greater than or equal to its required resources S_i, then it is linked to the UAV. Here $a_{i,j} = 0$ indicates that a task is executed on its own MN and $a_{i,j} = 1$ shows that a task is offloaded on UAV j for execution. Consequently, the related restriction for MN i, as well as the allocated resources, can be adjusted to $r_i \geq a_{i,j}S_i$. The deterministic UAV's optimal deployment is obtained by optimizing the total of the potential rates of the MNs as well as ensuring fair UAV coverage, with a fixed $\{S_i|\forall\, i\}$. The task needs to be transmitted to the UAV and computed by the UAV's MEC server, for the task assigned to the UAV. The result is returned to the MN upon execution. The distance of MN i to UAV j is

$$d_{i,j} = \sqrt{(x_i - X_j)^2 + (y_i - Y_j)^2 + H^2}, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{4}$$

The distance between two UAVs is:

$$d_{j_1,j_2} = \sqrt{(X_{j_1} - X_{j_2})^2 + (Y_{j_1} - Y_{j_2})^2}, \quad \forall j_1, j_2 \in \mathcal{J}, j_1 \neq j_2 \tag{5}$$

If a task is executed on a UAV, the MN should be within the coverage area of the UAV. Under the restrictions of computation offloading and resource allocation, the following constraint should be satisfied for fair coverage of the MNs by the UAV:

$$\mathcal{C}_1 : a_{i,j}d_{i,j} \leq R_j, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{6}$$

Here, the constraint $\mathcal{C}_1$ indicates that if a task is transferred to UAV j, MN i should be located in the coverage area of UAV j. R_j is the coverage radius of each UAV and the UAVs are assumed to have directional antennas with a fixed beamwidth θ. The coverage radius of each UAV is $R_j = H \cdot tan\theta$, and the UAVs fly at a constant altitude H [31]. In our system, there are three models: the UAV hover model, the local execution model, and the MEC execution model.

3.1 UAV Hover Model

The energy required by a UAV to hover at a fixed location for a period of time [32] can be represented as

$$E^H = P_0 T_h \tag{7}$$

where the hover time and hover power are denoted by T_h and P_0 respectively.

3.2 Local Execution Model

For the computing ability of MN $i \in \mathcal{I}$, the number of CPU cycles per second of MN $i \in \mathcal{I}$ is represented as l_i. In addition, the delay-tolerant computing tasks carried by $i \in \mathcal{I}$ at time slot t are represented as $Z_i(t) \triangleq (D_i(t), C_i(t))$, where $D_i(t)$ represents the size of the input data for computation and needs to be measured in bits similar to program codes, and $C_i(t)$ represents the total number of CPU cycles required to accomplish these computing tasks [33]. If a task is sent to UAV j, MN i should be positioned inside UAV j's coverage area. When an MN performs a calculation task locally, it must assign computing resources to the task. The computation capacity allocation matrix is written as $\mathbf{L}$, whose element $l_{i,t}$ indicates the computing capacity of the MN i in Hz, at time slot t, where $i \in \mathcal{I}$ [33]. The required time for an MN i to complete a computational task locally is given as follows,

$$T_{i,t} = \frac{C_i}{l_{i,t}} \tag{8}$$

It is essential that the computing resource allocated to the MN by each MEC server does not surpass the overall resource capacity. The amount of energy used to perform operations locally is determined by the number of required computer cycles [33], expressed as follows:

$$E_{i,t} = q_0 l_{i,t}^2 D_i C_i \tag{9}$$

where $q_0 > 0$ represents the operating coefficient of capacitance.

3.3 MEC Execution Model

The MNs first transmit the input parameters necessary for computation to the MEC server on the UAV before offloading the computational tasks to the system. The MEC servers calculate the tasks in terms of the received data. The MEC servers return the results to the MNs when the calculations are completed. It is worth mentioning that the amount of data in the outputs is significantly less than the input data. Therefore, the energy usage and task execution time during the phase of returning the results are not discussed in this work. In addition, the time required to compute a task on a MEC server is split into offloading and execution time [32]. The total time to complete a task includes both transmission and computing time on UAV j, i.e.,

$$T_{i,j} = \frac{D_i}{r_{i,j}} + \frac{C_i}{F_{i,j}}, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{10}$$

where $F_{i,j}$ represents the computing resources assigned to the task on the UAV j, and $r_{i,j}$ reflects the uplink data rate. The required energy to complete a task on the MN locally is determined as follows:

$$E_i^{MN} = \eta_c (F_i)^{v-1} C_i, \forall i \in \mathcal{I} \tag{11}$$

where η_c represents the effective switching capacitance and v represents a positive constant. In order to make the application more realistic [32], we set v = 3 in this paper. Furthermore, the overall energy used to complete a task

is the sum of the transmission and computation energy of UAV j as follows:

$$E_{i,j}^{UAV} = P\frac{D_i}{r_{i,j}} + \eta_c(F_{i,j})^{v-1}C_i, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{12}$$

where P shows the transmission power of each MN.

The transmission delay and transmission time on the downlink can be omitted since computing tasks often use smaller data sizes than offloaded tasks, and downlink transmission rates are considerably greater than uplink transmission rates [34]. As a result, the total transmission and computation delays are equivalent to the overall offloading delay, as follows:

$$d_{i,j,t}^{overall} = d_{i,j,t}^{tra} + d_{i,j,t}^{comp} \tag{13}$$

where $d_{i,j,t}^{tra}$ is the transmission delay, determined by dividing the throughput by the transmission rate, and $d_{i,j,t}^{comp}$ is the computation delay. The throughput of MN i at time slot t is computed [35] as

$$K_{i,j,t} = min\left(Q_i(t), \tau B_j log_2 \frac{1 + p_t h_j}{\delta^2}\right) \tag{14}$$

where the task queue of MN i is $Q_i(t)$. B_j and h_j represent the bandwidth and uplink channel gain of the subchannel UAV j, the noise power is denoted by δ^2, and p_t refers to the transmission power at the time slot of τ. Therefore, $d_{i,j,t}^{tra}$ can be computed as follows:

$$d_{i,j,t}^{tra} = \frac{K_{i,j,t}}{D_i(t)} \tag{15}$$

where $D_i(t)$ refers to the quantity of data sent to the edge server. According to the computational intensity model in [34], the processing density of the transmitted data for the allocated task is the number of CPU cycles required for completing each bit of data in the computing task, which is defined as ρ (CPU cycles/bit). In addition, $K_{i,j,t}$ CPU cycles are required to process the data of the subchannel UAV j. Denote the number of computing tasks performed at MN i as $d_i(t)$ over time slot t, the computational delay d^{comp} can be calculated as:

$$d_{i,j,t}^{comp} = \frac{K_{i,j,t} \times \rho}{d_i(t)} \tag{16}$$

Next, we explain the main optimization objective. In order to reduce the system's energy consumption, which includes the energy required to accomplish all tasks in the local computing pattern or the computation offloading pattern as well as the energy required for UAVs to hover, we must jointly optimize the computing resource allocation and computation offloading. The goal of resource allocation is to reduce the system's energy usage and computation time at each UAV location (X_j, Y_j, H). The joint optimization problem for computing resource allocation and computation offloading is described as:

$$\min_{N, X_j, Y_j, a_{i,j}, F_{i,j}} \sum_{i=1}^{I}\left(a_{i,j}E_i^{MN} + \sum_{j=1}^{J} a_{i,j}E_{i,j}^{UAV}\right) + \beta N E^H \tag{17}$$

subject to the following constraints:

$$\mathcal{C}_2 : d_{j_1 j_2} \geq d_{min}, \forall j_1, j_2 \in \mathcal{J}, j_1 \neq j_2 \tag{18}$$

$$\mathcal{C}_3 : \sum_{i=1}^{I}\sum_{j=1}^{J} a_{i,j} \leq n_{max}, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{19}$$

$$\mathcal{C}_4 : a_{i,j}T_{i,j} \leq T, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{20}$$

$$\mathcal{C}_5 : F_{i,j} > 0, \forall a_{i,j} = 1, i \in \mathcal{I}, j \in \mathcal{J} \tag{21}$$

where the constraint $\mathcal{C}_2$ indicates that a minimum distance $d_{j_1 j_2}$ must be maintained between any two UAVs to avoid the collision. The constraint $\mathcal{C}_3$ indicates that due to restrictions in the computational capability of the MEC server, a maximum number n_{max} of tasks can be executed by each UAV. The delays for each task are indicated by the constraint $\mathcal{C}_4$. Based on a previous study [32], we assumed that the task's output could be returned to the MN with a low transmission delay. As a result of the restriction $\mathcal{C}_5$ if the task runs in UAV j, the computing resource assigned to this task is larger than 0; otherwise, it is equal to 0. In this work, β is the weight coefficient that is set empirically to 1 in our experiments, in terms of tests with values such as 0, 0.5, and 1. Because this system included both MNs and multi-UAV-enabled MEC, the goal of this system is to improve the UAV deployment and task computing to reduce the system energy usage. This includes energy from the local computational or MEC computation for performing all tasks, as well as the energy for the hovering of the UAVs. In Section 3.3, it is clear that (17) is a non-convex nonlinear optimization problem. Therefore, it cannot be addressed using traditional optimization methods. The cooperative evolutionary computation techniques have the potential to address it since they are a kind of population-based heuristic search methods that do not need gradient information. We present a cooperative evolutionary computation technique to find both locally and globally optimal solutions to the optimization problem to reduce computational time and energy consumption.

4 Optimized Computing Resource Allocation (OCRA)

In this section, we present a two-layer OCRA model for cloud-based communication with adaptive computing resource allocation. This model constitutes the first layer responsible for calculating the priority of the subcomponents and distributing the resources appropriately. Based on an approach for calculating the contribution periodically, an algorithm is devised to efficiently distribute the resources. As the second layer, a pool model is designed to ensure the full use of unbalanced resource, as illustrated in Figure 2. The arriving tasks are ordered within the task buffer at the MN and handled on a first-in-first-out (FIFO) basis in this algorithm. They are carried out locally at the MN or transferred to the UAV-enabled edge server. As can be observed, task execution at the MN and task offloading at the UAVs are connected and cannot be separately solved.

To addressing the optimization problem, a cooperative evolutionary algorithm is proposed which takes two steps, namely, decomposition and optimization. In the first step, a problem is decomposed into sub-problems.

Then, an optimizer (MN / UAV solution) is assigned to each sub-problem. This process is applied to the widely known dimension-distributed model [36]. The decomposition step is executed based on the distributed optimization method [36]. In this step, M-dimensional problem $Z = \{z_1, z_2, \cdots, z_M\}$ can be solved to prevent excessive resource imbalance between UAV processors. Meanwhile, for serial cooperative coevolution, when one MN population evolves in computing the task locally over time, the other UAV solution remains static, which is defined as

$$\begin{aligned} & Z = Z_1 \bigcup Z_2 \bigcup \cdots \bigcup Z_M; \\ \text{where} \quad & \forall p \in [1, M], Z_p \neq 0; \\ & \forall p, q \in [1, M] \wedge p \neq q, Z_p \bigcap Z_q = 0 \end{aligned} \tag{22}$$

In the optimization step, the fitness of each MN population is assessed via integration and comparison with the other best entities. The best entity in the pth population and the qth generation is presented as best solution b_p^q. The fitness of a MN population pop^p is defined as follows:

$$\begin{aligned} & F(pop^p) = F\left(pop^p, \overline{Z_p^q}\right) \\ \text{where} \quad & \overline{Z_p^q} = \left(b_1^q, \cdots, b_{p-1}^q, b_{p+1}^q, \cdots, b_M^q\right) \end{aligned} \tag{23}$$

Each MN population in each generation should update its own tasks until the update of the corresponding subcomponent is complete and the global best (GB) solution for allocating the limited resources from the idle subcomponent is obtained.

$$GB = \left(b_1^q, \cdots, b_{p-1}^q, b_{p+1}^{q-1}, \cdots, b_M^{q-1}\right) \tag{24}$$

The resource allocator in the cooperative evolutionary model gathers the contributions of all the UAV processors in each iteration for reallocation. The basic concept is to allocate the limited resources from the idle processor to the system. As we can see in Figure 2, the sub-process determines the contribution of each solution to the achievement of the overall goal. It starts at 0 and grows with time. Thus, the best member is denoted as $best_p^{q,g}$, where p, q and g refer to the population, generation, and iteration $(g \geq 1)$, respectively. The subcomponents (e.g. UAVs) are chosen based on their ability to improve the overall fitness. As a result, the resource allocator will select a subcomponent that contributes more to the total fitness. The contribution of the UAV processor for serving MN population p is shown as ϕ_p and it is determined as follows:

$$\begin{aligned} & \Delta z = F\left(b_p^{q,g-1}, \overline{Z_p^q}\right) - F\left(b_p^{q,g}, \overline{Z_p^q}\right) \\ & \phi_p \leftarrow \phi_p + \Delta z. \end{aligned} \tag{25}$$

The resource allocator then adjusts the allocation. A subcomponent is allocated to the MN population that has a higher priority. The host process calculates the priority (ψ_p), which is defined as

$$\psi_p = \frac{\phi_p}{\lambda^{p,g}} \tag{26}$$

This formula indicates the involvement of the subcomponent where $\lambda^{p,g}$ shows the subcomponent in the gth iteration for serving the population p and ϕ denotes the contribution made by a subcomponent. Furthermore, the maximum and minimum bounds are adjusted to prevent extreme imbalance in resource allocation and limit the maximum and minimum number of processors for a subcomponent, represented as mab and mib, respectively.

Algorithm 1 Resource Allocation Algorithm

Input: Number of Pool $\{Pool_1, Pool_2, \cdots, Pool_M\}$, minimum and maximum bound mib and mab, number of sub-components M.
$\psi = 0_{1\times M}$;
Output: Optimal resource allocator with min $f(x)$ ▷ *$f(x)$ is objective function, see (27).*
for all $Pool_p \in Pool$ **do**
    Calculate ψ_p according to (26);
for p= 1 to M **do**
    for q= $p+1$ to M **do**
        if $Pool_q.\psi < Pool_p.\psi$ **then**
            swap ($Pool_p$, $Pool_q$)
    end
end
$pr = -1$ ▷ *pr is the index of a pool that needs more processors*
$pd = -1$ ▷ *pd is the index of a pool that can allocate its processors to other pools*
$p = M$
while $|\lambda^p| \geq mab$ & $p \geq 1$ **do**
    $p = p - 1$
$pr = p$
$p = 1$
while $|\lambda^p| \leq mib$ & $p < pr$ **do**
    $p = p + 1$
$pd = p$
if $pd \neq -1$ **then**
    $Pool_{pd}$ denotes a processor to $Pool_{pr}$
end
end

The efficient UAVs that facilitate task computation are defined as UAV-solutions, $\mathcal{X} = \{x_1, x_2, ..., x_J\}$. The MN-solutions are assessed by utilising an objective function that is dynamically updated based on its role in the generation of efficient UAV-solutions in the UAVs' population. On the contrary, each UAV-solution is assessed by pairing it with the MN-solution in the MNs' population, as the best match for the UAV-solution. We define the following objective function to evaluate each UAV-solution $\mathcal{X}$ based on the computation operation time,

$$minf(x_j) = \sum_j^J \omega_j (T - \tau_j) \tag{27}$$

where T is the mission completion time, τ_j is expected end time for computation operation and ω_j is the weight of a maximum number n_{max} of tasks that can be executed by each UAV, subject to $\sum_j \omega_j = 1$. In the network, we set each ω_j according to the highest flow transmitted through the MN. We update the weights as the algorithm evolves, depending on the feedback from the evolution of the MN solutions.

The resource allocation pseudo code is summarized in

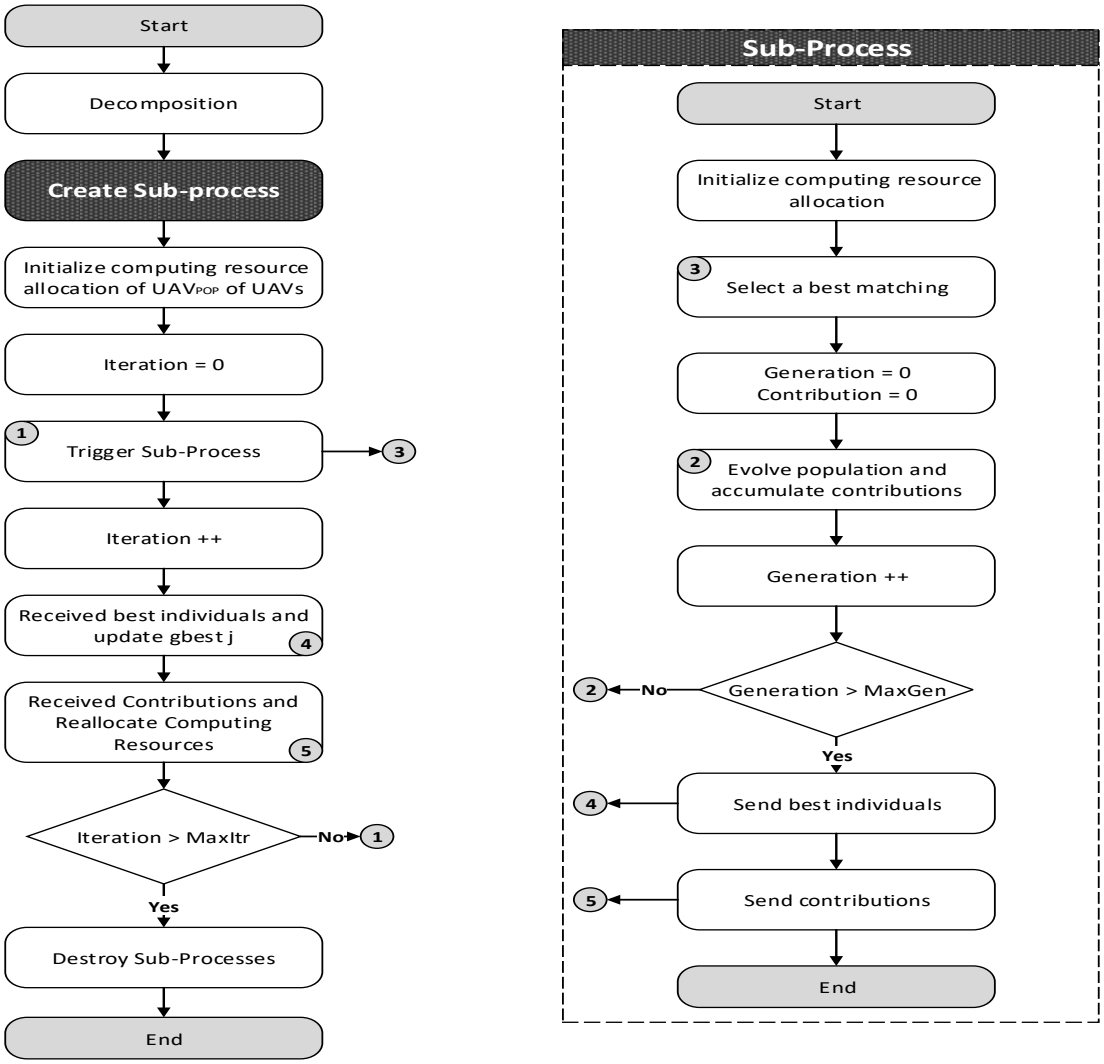

Figure 2: Cooperative evolutionary flowchart.

Algorithm 1. First, the MN solutions are created randomly and then the UAV solutions are generated constantly. Each population (i.e. 1 to M) can use some processors. In order to efficiently distribute and allocate the UAV processors to populations, it calculates the priority of populations (Lines 4-5). Then, it sorts the populations based on the priority in an ascending order (Lines 6-9). The two indices pr and pd are initially set to -1 (Lines 12, 13). It looks for a population with high priority that has processors less than mab (Lines 14-16) and sets pr with the index of this population (Line 17). It also looks for a population with a low priority which the number of its processors is higher than mib (Lines 18-20), and then sets pd to the index of this population (Line 21). Finally, a processor from the population pd will be incorporated into population pr (Line 23).

The OCRA resource allocator is capable of compiling all the UAV processor contributions in each iteration, and then reallocating the processors as necessary. Its basic concept is to transfer limited computational resources from stagnant UAV processors that contribute less to those that contribute more. The contribution of each UAV processor to the overall goal is computed in the relevant subprocess. It starts off at zero and grows with each generation. The contribution of the UAV processor is defined as Λ_j in Equation (28). After that, the resource allocator moderately modifies the allocation. If two UAV processors are chosen each time, the one with a lower priority gives the other one a processor. The host process, defined in Equation (29), determines the priority by accounting for the contribution made by each individual processor. In addition, a lower bound and an upper bound, termed as mib and mab, respectively, are specified to limit the lowest and the maximum number of processors that a UAV processor can hold to prevent excessive resource imbalance.

4.1 Pool Layer

The performance and convergence behaviour of most EA algorithms can be affected by two parameters, namely, the size of the population and the number of generations. In this work, a verification plan is implemented inside this layer to assist the optimizer in making effective use of a given computational resource. Consequently, the cooperative evolutionary model not only enables the scalability in coping with high-dimensional populations but also allows the UAV processors to successfully adapt to resource allocation changes and utilize the computing resource allocated to

them efficiently. The population plan modifies the population of an optimizer, whereas the generation plan modifies the number of generations. A validation plan describes how the evolution approach and the number of processors it owns are matched. When the number of MN populations allocated to a UAV processor changes, the "unverified" situation occurs.

Definition 1: Consider a pool λ owned by the same processor i.e. the jth processor in the qth generation, is defined as $\lambda^{j,q}$. An unverified situation occurs if and only if $\lambda^{j,q} \neq \lambda^{j,q+1}$ and the verification plan of $\lambda^{j,q}$ in the $(q+1)$th generation is not executed. When a subprocess detects an unverified situation, it instantly applies a verification plan to fix it in order to fully utilize the computational resources available to it.

As a population-distribution technique, the pool model controls the population by constructing a shared pool. Figure 3 elucidates the structure of the OCRA model. The whole resource pool, which includes all the subcomponents (i.e. UAV processors), is accessible to all MNs. On the other hand, each processor is limited to updating only one part of the pool at a time. The pool is built as an array of N UAV processors that is divided into M MN populations. The pooling technique has the advantages of flexibility and scalability. Adding extra processors is as simple as increasing the pool size or rearranging the segments, because all processors share the same resource pool. In the OCRA paradigm, these features would facilitate dynamic allocation of resources. Each task Z_p is associated with an MN population S_p in terms of the population distribution. Each MN and its assigned UAV processor are combined to build a pool layer in the OCRA model,

$$\Lambda_j = (S_p, \lambda_j) \tag{28}$$

where λ_j refers to UAV processors which are assigned to S_p. According to the pool model framework, the N UAV processors are defined as $\{\lambda_1, \lambda_2, \cdots, \lambda_N\}$. For example, two UAV processors denoted as $\{\lambda_1, \lambda_2\}$ are allocated to the first MN population S_1. Three UAV processors denoted as $\{\lambda_3, \lambda_4, \lambda_5\}$ are allocated to the second MN population S_2. In general, the MN populations are synchronised at a predefined number of generation intervals. During synchronisation, the resource allocator also records the role of each UAV processor to achieve the global objective.

4.2 Queue Based Offloading

In a queue-based architecture, the MN computation tasks are based on the data bits of each task computed by MN i at time slot t and it is expressed as $A_i(t)$. The arrived tasks are dispersed independently with an average rate of r_i at distinct time windows. Each MN can keep up with a queue of arriving tasks for a certain amount of time. The MNs employ a task buffer, where the tasks are queued and processed in the order of arrival time. The procedure might be performed locally at the MN or transferred to the UAV. The local and offloaded tasks from MN i are denoted by $m_i(t)$ and $o_i(t)$, respectively. We define the processing density as the number of CPU cycles required to complete a one-bit compute task and it is denoted as ρ. The amount of data stored at local queue at the MN i is $m_i(t)$, defined as

$$m_i(t) = \frac{C_i(t)\tau}{\rho} \tag{29}$$

where $C_i(t)$ is the number of CPU cycles of MN i at time slot t with slot length τ. The task queue backlog of MN i at time slot t i.e. $Q_i(t)$ can be updated as:

$$Q_i(t+1) = max\left\{Q_i(t) - m_i(t) - o_i(t), 0\right\} + A_i(t) \tag{30}$$

In case of task offloading, the UAV processor needs to store the task in a large number of parallel buffers. Because the UAV processor often has greater computation capacity, it can handle offloaded tasks. For MN i, the expansion of the queue length $L_i(t)$ at the UAV processor will be as follows:

$$L_i(t+1) = max\left\{L_i(t) - m_i(t), 0\right\} + o_i(t) \tag{31}$$

The current incoming tasks should be saved in the queue buffers and completed when a time slot becomes available. The inequalities $Q_i(t+1) \geq A_i(t)$ and $L_i(t+1) \geq o_i(t)$ are defined for the MNs within the cell. The length of the queues and the value of the selected indicators are both set to zero during the initialization phase. After that, the decision-making procedure is carried out one slot at a time. At the start of the t-th slot, the UAV processor calculates the value of the weighted total throughput, $\theta_{i,j,t}$.

The channel selection indicator is denoted as $x_{i,j,t}$. The best choice for UAV j can be the one with the smallest computing cost $O(M)$, see section(4.3). Subsequently, UAV processor j updates all queues and sets $\theta_{i,j,t} = 1$. The iteration continues in the following slot until $t > T$. The UAV processor j uses two main metrics to make decisions in each time slot. The first term is $\bar{\theta}_{i,j,t-1}$ that shows the estimation of $\theta_{i,j,t}$ and allows the system to choose the previously determined optimal choice in each time slot. The second term is the confidence bound $\widehat{x}_{i,j,t-1}$, which is employed to strike a balance between exploration and exploitation.

This term enables MN i to evaluate solutions with restricted decisions in order to improve the estimation accuracy. The term $\widehat{x}_{i,j,t-1}$ indicates how many times i has chosen the j-th option over the slot t. The throughput $\widetilde{\theta}$ is the estimation of θ in the slot t for option j as follows:

$$\widetilde{\theta}_{i,j,t} = \bar{\theta}_{i,j,t-1} - \omega\sqrt{\frac{2\ln t}{\widehat{x}_{i,j,t-1}}} \tag{32}$$

where the first term on the right-hand side of the equation shows the performance of option j, and ω reflects the weight of exploration against exploitation, with a greater value of ω indicating a stronger preference for exploration. The minimum estimation value after estimating $\widetilde{\theta}_{i,j,t}$ for all subchannel UAV j is defined as follows

$$j = \arg\min_{j}\left\{\widetilde{\theta}_{i,j,t}\right\} \tag{33}$$

Accordingly, $\widehat{x}_{i,j,t}$ is updated as

$$\widehat{x}_{i,j,t} = \widehat{x}_{i,j,t-1} + x_{i,j,t} \tag{34}$$

Algorithm 2 demonstrates how to build queues, make

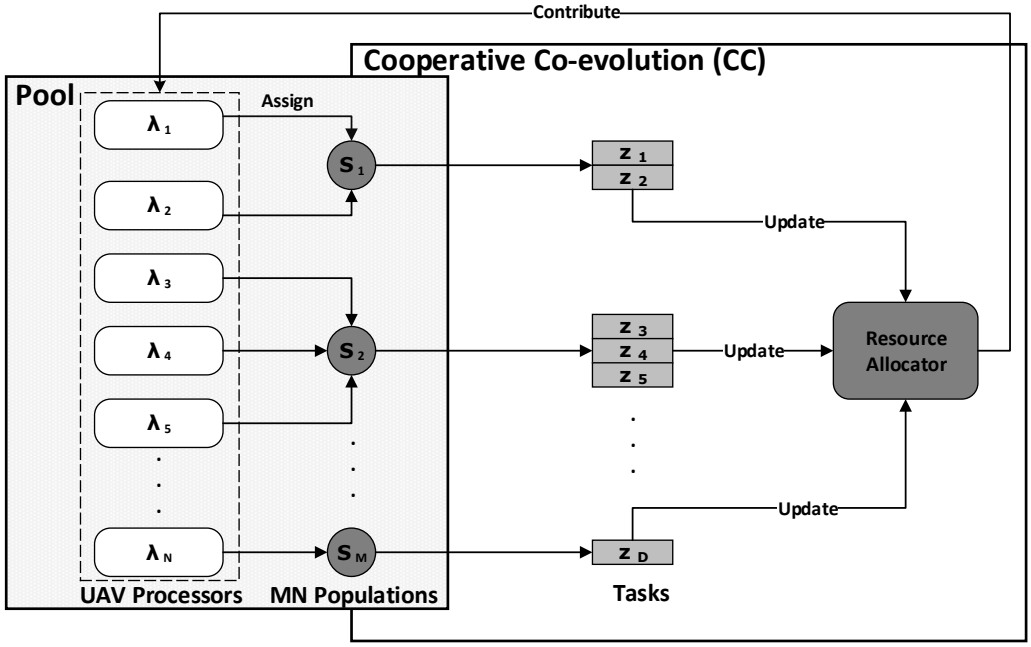

Figure 3: Structure of the OCRA model.

Algorithm 2 Queue Algorithm

1: **Input:** ω; τ_j; $\theta_{i,j,t}$; $x_{i,j,t}$
2: **Output**: Optimal offloading decisions $\widetilde{\theta}_{i,j,t}$
3: //Step 1: Initialization;
4: Set the initial amount of data; $\bar{\theta}_{i,j,t} = 0$, $\widehat{x}_{i,j,t} = 0$;
5: **repeat**
6: //Step 2: Decision making;
7: Compute the estimation value of the MN i towards option UAV j as (32);
8: Select the optimal option j based on (33);
9: Update $\widehat{x}_{i,j,t}$ as (34);
10: Update $m_i(t)$ as (29);
11: Update computation task executed at MN as (30);
12: Update computation task executed at edge server as (31);
13: **until** $t > T$

task offloading decisions, and update the queues. The UAV processor j chooses the task offloading option with the maximum probability of success. In an ideal case, the processor j has the entire current information, which includes both local and non-local data. The lines 6–12 are repeated until $t > T$ in the system. Algorithm 2 is a low-complexity algorithm for dealing with the sequential decision-making problem. In each time slot, MN makes decisions based on only two kinds of local information: $\bar{\theta}_{i,j,t-1}$ and $\widehat{x}_{i,j,t-1}$ where $\bar{\theta}_{i,j,t-1}$ represents the empirical estimation of $\theta_{i,j,t-1}$ up to slot t, and $\widehat{x}_{i,j,t-1}$ represents the number of times that has selected the j-th option up to slot t.

Algorithm 2 can manage the computation tasks based on the queueing and transmitting (or processing) tasks. This algorithm can adapt to the variations in the amount of data backlog and service state due to the endowed context awareness, which is achieved through the dynamic adjustment of the channel selection strategy. The UAV processor can store the task in many parallel buffers to minimize the task execution latency. It can handle offloading tasks and increase the data processing rate.

4.3 Model Complexity

The cooperative coevolution can be easily parallelised due to its population-based structure, which is one of the advantages of the proposed algorithm. In general, the most time-consuming aspect of an evolutionary algorithm is fitness evaluation, which allows parallel processing of multiple individuals. Because the proposed model is based on cooperative coevolution, it employs a superior parallelisation technique by allocating a processing thread to each population rather than each individual. As a result, the number of context switching and thread scheduling is reduced. We analyze the complexity of the proposed algorithms in this section.

Algorithm 1: The computational complexity of the priority computation is $O(M)$ and the complexity of sorting is $O(M^2)$. Additionally, in the worst case, the computational complexity for assigning the processor to a pool is $O(M-1)$. Hence, the computational complexity of this algorithm is approximately $O(M) + O(M^2) + O(M-1)$.

Algorithm 2: The computational complexity of Algorithm 2 includes three main parts, which are initialization, estimation value calculation, and optimal value selection and renewing queues with the complexity of $O(3M+6)$, $O(M+1) + O(M)$, and $O(2M+5)$, respectively. Thus, the total computational complexity of Algorithm 2 is approximately $O(3M+3) + O(M+1) + O(M) + O(2M+5)$.

5 Numerical Evaluation

In this section, we present the simulation results to show the performance of the proposed algorithm. We consider a UAV-enabled MEC (UAV-MEC) system with 10 servers placed at

Table 1: List of Simulation Parameters.

Description	Parameter	Value
System bandwidth	B	40 MHz
UAV transmission power	P_t	100 dBm
Simulation area	–	$10\ \mathrm{km}^2$
Number of macro cell base stations	MBS	100
Number of APs configured around each cell	Aps	20
Radio range of the UAVs	R	500-800 m
Power noise at the UAV	σ^2	10^{-9} W
Noise power spectral density	N_0	$174\ \mathrm{dBm/Hz}$
Number of time slots	N	50
UAV altitude	H	10 m
Total number of CPU cycles to accomplish tasks	C_i	50
Effective switching capacitance	η_c	10^{-28}
Channel power gain	g_0	-50 dB
Probabilistic SINR Threshold	SINR Threshold	-6 dB
Size of computation task input data for ith MN (bit)	$\bar{D}$	150 Mb
Maximum number of mobile nodes	MN	100
Receiving threshold	Th	1.17557e-10 W
Radio propagation model	-	Friis model
Antenna type	-	Omni Antenna
Maximum data rate (WiMAX)	-	1882 Kbps
PHY Mode	-	256 OFDM
Maximum data rate (UMTS)	-	384 Kbps
Constant speed of the UAV	V_c	10 m/s

various locations and 100 MNs randomly distributed within a 2-D squared area of 1000 x 1000 m^2. Our system settings involve the interactions between the MN devices, UAV, and ECs, which distinguish our study from prior studies. We used Friis free-space propagation model [37], which works in the far-field region. We utilised OMNeT++ and MATLAB as the two main platforms. We used the LiveLab dataset [38] to evaluate our approach. LiveLab is a system developed at Rice University to assess real-world smart-phone usage and wireless networks by using a reprogrammable in-device logger for long-term MN research. Data from various devices, such as phones and tablets, are included in the dataset. Table 1 presents the setup of the important parameters for evaluating the performance of the proposed method. The performance of the proposed scheme is compared with the following benchmark schemes:

1) Straight-line flight: The scheme optimizes task computation and resource allocation by flying the UAV directly from its starting position to its destination at a constant speed V_c, similar to a basic straight-line flight trajectory [39]. The resource allocation is determined by the proposed OCRA model in Section 5.

2) Equal offloading time allocation: The proposed OCRA model is used to describe the computation resource allocation and completion time for the optimal resource allocation and equal offloading time allocation [40]. This method defines the assigned slot time t for the i-th device to offload the computing tasks to the UAV.

3) Full offloading: The scheme provides optimal resource allocation and computation offloading without local computing at the ground devices, similar to [40]. The arriving tasks are offloaded to the UAV for processing.

4) Hybrid (UAV-MEC)-MN scheme: The UAV processes half of the tasks, and the MNs process the other half [41]. We call our suggested approach collaborative UAV-MN scheme. The proposed scheme is compared with the UAV-only, MN-only, and hybrid UAV-MN schemes to demonstrate the benefits of leveraging the computing resources at both the UAV and MNs. To this end, the performances of our proposed OCRA model is compared with the existing benchmarks and hybrid (UAV-MEC)-MN schemes.

Figure 4 shows the relationship between the completion time and energy consumption of the UAV with $D_i(t)$= 150 Mb. It is first observed in Figure 4 that the tradeoff between the energy consumption of the UAV and the completion time follows a "U" shape. The Pareto boundary is represented by the left side of the curve in Figure 4. The Pareto-optimal solution can achieve a balance between the two objectives. The curve is given by Eqs (14) and (15) for any t, $t \geq t_{min}$, where t_{min} is the minimum completion time for executing computation tasks $D_i(t)$. The energy consumption by the UAV initially decreases and gradually increases with the increase in the completion time. The reason for this is that the UAV's speed reduces as the completion time increases. As a result, the UAV's propulsion power consumption fluctuates based on its current speed. In the beginning, the decrease in the propulsion power consumption is more than the increase in the completion time. Then the energy consumption of the UAV decreases until the minimum value has been achieved. Next, the reverse situation happens, and then the energy consumption of the UAV increases. Figure 4 illustrates the difference between the UAV's energy consumption and completion time. Unlike the current MEC system that applies a method for the minimization of the energy used by the UAV, the proposed model may provide a solution to minimise the energy consumption of the UAV, either without a specified completion time or within an ideal completion time. The optimal completion time (approximately 2.1 s) that minimizes the energy consumption of the UAV in Figure 4 is also obtained.

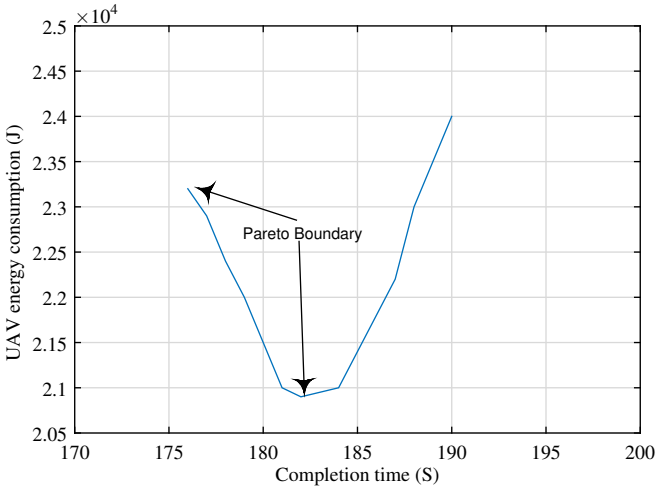

Figure 4: Completion time versus energy consumption.

Minimization of UAV energy consumption and execution time are two major problems in the UAV-enabled MEC system and they are both relevant to the speed of UAV flight. The formula $\int_0^T k_i(t)P_t dt$ refers to the UAV's energy consumption, where P_t is the transmission power for the MN at task offloading. The computation energy consumption of MN i is measured as $\int_0^T q_0(F_{i,k})(t)dt$, as in Eqs (14) and (15), where $q_0 > 0$ represents the effective capacitance coefficient of MN i. Therefore, we have $\int_0^T q_0(F_{i,k})(t)dt + \int_0^T k_i(t)P_t dt \leq E_i^{max}$, where E_i^{max} is the energy budget of MN i in pattern k. The MNs are randomly distributed in a rectangular area in the proposed UAV-enabled MEC system. The computational task input size, energy budget, and transmission power are considered identical for all MNs such that $D_i(t) = 150\ Mb$,

This article has been accepted for publication in IEEE Transactions on Aerospace and Electronic Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAES.2023.3251967

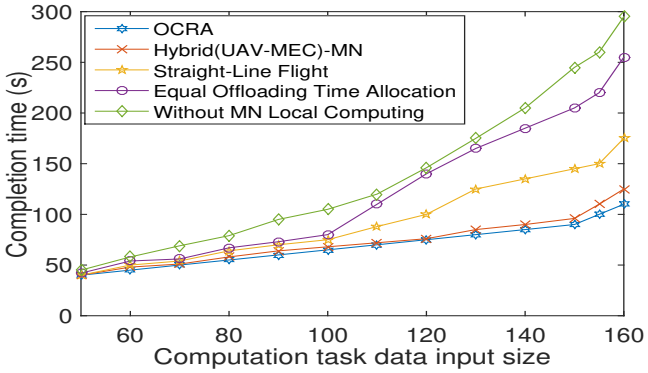

Figure 5: Completion time vs. task-input data size $\bar{D}$.

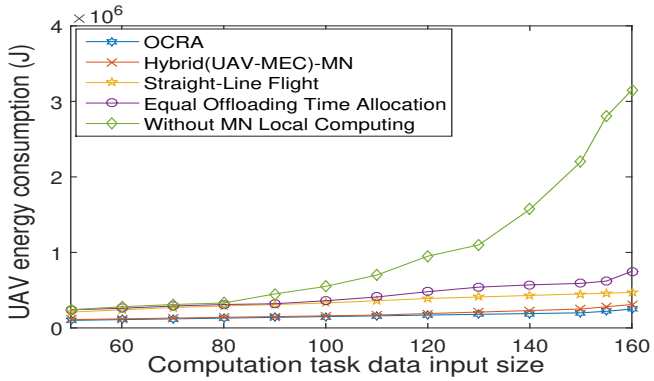

Figure 6: Energy cons. vs. task-input data size $\bar{D}$.

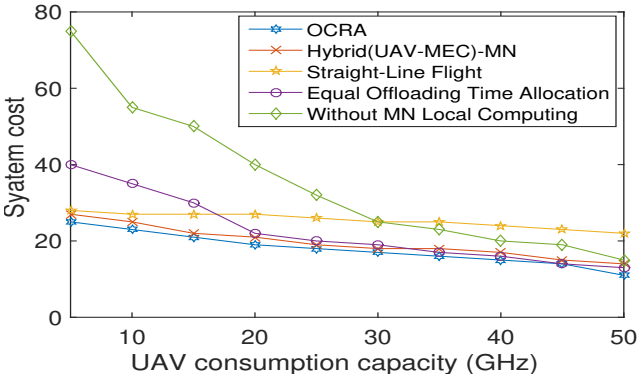

Figure 7: UAV computation capacity versus system cost.

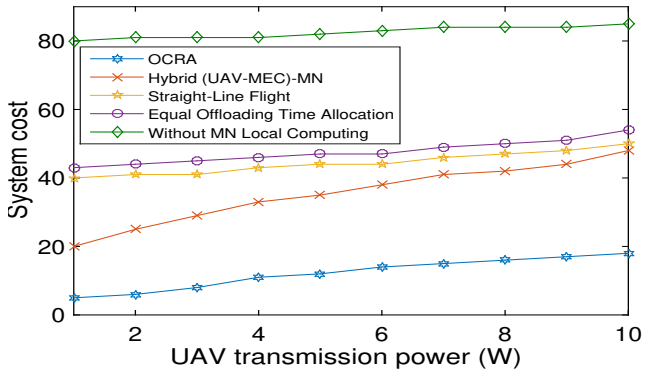

Figure 8: UAV transmission power versus system cost.

$E_i^{max} = 1\ J$, $P_t = \bar{P} = 0.1\ W$ for all i. All system parameters are listed in Table 1.

Figure 5 shows a comparison between different completion times. It can be seen from Figure 5 that the execution time of the proposed method increases linearly, while the execution time of the no MN local computing design, straight-line flying method and hybrid method increases exponentially. Although the execution time of all methods are almost the same at first, the execution time of other methods rapidly exceeds the execution time of the proposed method as the size of the input data increases. Therefore, for large-scale networks, applying the proposed method can significantly reduce the algorithm execution time and improve execution efficiency. It can also be observed that the completion time by the no MN cooperation design increases sharply with the increase in data size. The other designs achieve a smaller time compared with the no MN local computing design, this is because the MN as a helper can improve task computation. In addition, it can be observed that the proposed design outperforms the other designs due to the joint optimization for computation and resource allocation.

Figure 6 shows the energy usage by the UAV versus the input data size $D_i(t)$. The improvement in the system performance is more significant when $D_i(t)$ is larger. The difference in performance between our joint design scheme and the straight-line flying benchmark implies that the recommended trajectory designs are beneficial. The comparison of our combined design technique with a benchmark based on equal time allocation for offloading further proved the advantage of the proposed model in terms of the offloading time. Improved flexibility in computing resource allocation can be achieved when the offloading time allocation is adjusted.

The performance disparity between our cooperative design approaches and those without local computing benchmarks highlights the importance of local computing and UAV offloading integration. For the no MN design, high energy consumption is incurred for task computing, for the following main reasons. First, in this design, the MN does not help compute. Second, with increase in the task bits, the energy consumption increases with regard to the cube of the required task bits. Last but not least, due to the trajectory pattern being fixed, the straight flight design is limited in mobility exploitation compared with the proposed method, which leads to larger energy consumption.

Figure 7 illustrates the influence of system cost when the UAV processing capacity increases from 4 to 50 GHz. According to [42], the computing capacity of the edge server is usually 2–5 times that of the MNs. Therefore, the computing capacity of the edge server is set to 3 times that of the MNs. The system cost of the proposed method does not vary when the computation capacity changes, this is because UAVs are able to provide higher-quality computing or offloading services with less flight time when the battery capacity has been improved [43]. This leads to energy consumption minimization. We also observe that when the computing capacity increases, the system costs of all three offloading techniques decrease because more computation resources are available at the UAV level to reduce the task processing delay.

Finally, we consider the effect of system cost as the

transmission power of the UAV is increased from 1 to 10 W, as shown in Figure 8. The system cost of the UAV-only strategy increases as the transmission power of the UAV increases, since this design requires all MNs to offload all tasks to the UAV for execution without local computation. We also find that the system cost increases with the increase in the transmission power for the other four techniques, because of the corresponding increase in the usage of the UAV downlink transmission energy. The main reason is that the proposed method uses optimal offloading decisions for task offloading in a queue-based offloading algorithm and also uses a pool layer in the resource allocation algorithm. The pool layer used to store the explored resource is also an important reason for improving the stability and performance over other algorithms. We find that our method outperforms the baseline schemes in terms of system cost reduction in all of the above scenarios, indicating the significance of UAV-EC collaboration in task offloading operations.

6 Conclusions and Future Directions

We have presented an optimized computing resource allocation (OCRA) model to make an effective use of the computational capability across the system. The main idea behind the proposed adaptive resource allocation method is to use the compliance rules of the cooperative coevolution technique to adjust the resource allocation. Our simulation results indicate that the algorithm for computing resource allocation and the population policy is effective. Moreover, the presented distributed architecture has higher efficiency and scalability. In the proposed edge computing framework, each UAV-enabled edge server satisfies the requirements of mobile device communication. In addition, the MN's accessibility rates can be significantly improved, and the energy consumption of the MNs can be reduced. Significant enhancement in the performance was obtained using the proposed designs when compared with the other baseline schemes. In the future, we will focus on cross-layer resource optimization at the edge server, including task assignment and resource allocation in the presence of uncertainty.

Acknowledgment

This research was sponsored by the US Army Research Laboratory and the UK MOD University Defence Research Collaboration (UDRC) in Signal Processing under the SIGNeTS project. It is accomplished under Cooperative Agreement Number W911NF-20-2-0225. The views and conclusions contained in this document are of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, the MOD, the U.S. Government or the U.K. Government. The U.S. Government and U.K. Government are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. For the purpose of open access, the authors have applied a creative commons attribution (CC BY) licence to any author accepted manuscript version arising. We would like to thank the reviewers and editor for their helpful comments in improving this article.

References

[1] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 7–12.

[2] S. Goudarzi, M. H. Anisi, H. Ahmadi, and L. Musavian, "Dynamic resource allocation model for distribution operations using SDN ," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 976–988, 2020.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] J. Wang, K. Liu, and J. Pan, "Online UAV-mounted edge server dispatching for Mobile-to-Mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1375–1386, 2019.

[5] H. Dai, H. Zhang, B. Wang, and L. Yang, "The multi-objective deployment optimization of UAV-mounted cache-enabled base stations," *Physical Communication*, vol. 34, pp. 114–120, 2019.

[6] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2018.

[7] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[8] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[9] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2010, pp. 300–309.

[10] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[11] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2013.

[12] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 42, no. 2, pp. 1–24, 2016.

[13] M. A. Ali, Y. Zeng, and A. Jamalipour, "Software-defined coexisting UAV and WiFi : Delay-oriented traffic offloading and UAV placement," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 988–998, 2020.

[14] C.-F. Liu, S. Samarakoon, M. Bennis, and H. V. Poor, "Fronthaul-aware software-defined wireless networks: Resource allocation and user scheduling," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 533–547, 2017.

[15] N. Zhao, X. Pang, Z. Li, Y. Chen, F. Li, Z. Ding, and M.-S. Alouini, "Joint trajectory and precoding optimization for UAV-assisted NOMA networks," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3723–3735, 2019.

[16] Y. Zhu, G. Zheng, and M. Fitch, "Secrecy rate analysis of UAV-enabled mmWave networks using matérn hardcore point processes," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 7, pp. 1397–1409, 2018.

[17] M. Chen, W. Saad, and C. Yin, "Liquid state machine learning for resource and cache management in LTE-U unmanned aerial vehicle (UAV) networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1504–1517, 2019.

[18] J. Lyu and R. Zhang, "Network-connected UAV: 3-D system modeling and coverage performance analysis," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7048–7060, 2019.

[19] S. Goudarzi, M. H. Anisi, D. Ciuonzo, S. A. Soleymani, and A. Pescape, "Employing unmanned aerial vehicles for improving handoff using cooperative game theory," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 2, pp. 776–794, 2020.

[20] P. Yang, X. Cao, X. Xi, Z. Xiao, and D. Wu, "Three-dimensional drone-cell deployment for congestion mitigation in cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9867–9881, 2018.

[21] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[22] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV -mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2017.

[23] M. N. Soorki, M. Mozaffari, W. Saad, M. H. Manshaei, and H. Saidi, "Resource allocation for machine-to-machine communications with unmanned aerial vehicles," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[24] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[25] H. Q. Le, H. Al-Shatri, and A. Klein, "Efficient resource allocation in mobile-edge computation offloading: Completion time minimization," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2513–2517.

[26] J. Ren, G. Yu, Y. Cai, Y. He, and F. Qu, "Partial offloading for latency minimization in mobile-edge computing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[27] U. Saleem, Y. Liu, S. Jangsher, and Y. Li, "Performance guaranteed partial offloading for mobile edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[28] M. Cui, G. Zhang, Q. Wu, and D. W. K. Ng, "Robust trajectory and transmit power design for secure UAV communications," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 9042–9046, 2018.

[29] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[30] M. Li, M. Huo, X. Cheng, L. Xu, X. Liu, and R. Yang, "Joint offloading decision and resource allocation of 5g edge intelligent computing for complex industrial application," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 1542–1547.

[31] 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 Transactions on Cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2019.

[32] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[33] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2016.

[34] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5225–5240, 2018.

[35] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, "Learning-based context-aware resource allocation for edge-computing-empowered industrial iot," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4260–4277, 2019.

[36] Y.-H. Jia, W.-N. Chen, T. Gu, H. Zhang, H.-Q. Yuan, S. Kwong, and J. Zhang, "Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 188–202, 2018.

[37] M. A. Sayeed, R. Kumar, and V. Sharma, "Efficient data management and control over wsns using sdn-enabled aerial networks," *International Journal of Communication Systems*, vol. 33, no. 1, p. e4170, 2020.

[38] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum, "LiveLab: measuring wireless networks and smartphone users in the field," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, pp. 15–20, 2011.

[39] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[40] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion time and energy optimization in the UAV-enabled mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7808–7822, 2020.

[41] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.

[42] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[43] J. Buczek, L. Bertizzolo, S. Basagni, and T. Melodia, "What is a wireless uav? a design blueprint for 6g flying wireless nodes," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, 2022, pp. 24–30.

Shidrokh Goudarzi is a lecturer in Computer Science at the School of Computing and Engineering at the University of West London, U.k. Prior to this, she was a research fellow at the Centre for Vision, Speech, and Signal Processing (CVSSP), University of Surrey. She was a senior lecturer at the Universiti Kebangsaan Malaysia (UKM). She received her Ph.D. degree in communication systems and wireless networks from the Malaysia-Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia (UTM). She received three-year full scholarship to study Ph.D. at (UTM). Then, she joined the Department of Advanced Informatics School at Universiti Teknologi Malaysia as a Postdoctoral Fellow from 2018 to 2020. She serves as a reviewer for some journals. Her research interests are in wireless networks, Artificial Intelligence, Machine Learning, Next-Generation Networks, Internet of Things (IoT) and Mobile/distributed/Cloud Computing.

Seyed Ahmad Soleymani is a research fellow at the Centre for Vision, Speech, and Signal Processing (CVSSP), University of Surrey. Previous to this role, he was a research fellow at the 5G&6G Innovation Centre, Institute for Communication Systems (ICS), University of Surrey. He received his Ph.D. in computer science from the faculty of engineering, UTM, Malaysia in 2019. His research interests are in Wireless Sensor Networks, Mobile Ad Hoc Networks, Vehicular Ad Hoc Networks, and UAVs Networks.

Wenwu Wang (M'02–SM'11) was born in Anhui, China. He received the B.Sc. degree in 1997, the M.E. degree in 2000, and the Ph.D. degree in 2002, all from Harbin Engineering University, China. He then worked in King's College London, Cardiff University, Tao Group Ltd. (now Antix Labs Ltd.), and Creative Labs, before joining the University of Surrey, UK, in May 2007, where he is currently a Professor in Signal Processing and Machine Learning, and a Co-Director of the Machine Audition Lab within the Centre for Vision Speech and Signal Processing. He is also an AI Fellow within the Surrey Institute for People-Centred Artificial Intelligence. His current research interests include blind signal processing, sparse signal processing, audio-visual signal processing, machine learning and perception, artificial intelligence, machine audition (listening), and statistical anomaly detection. He has (co)-authored over 300 publications in these areas including two books: Machine Audition: Principles, Algorithms, and Systems by IGI Global published in 2010, and Blind Source Separation: Advances in Theory Algorithms and Applications by Springer in 2014. His work has been funded by EPSRC, EU, Dstl, MoD, DoD, Home Office, Royal Academy of Engineering, National Physical Laboratory, BBC, and industry (including Samsung, Tencent, Huawei, Atlas, Saab, and Kaon). He is a (co-)author or (co-)recipient of over 15 awards including the 2022 IEEE SPS Young Author Best Paper Award, ICAUS 2021 Best Paper Award, DCASE 2020 Best Paper Award, DCASE 2019 and 2020 Reproducible System Award, LVA/ICA 2018 Best Student Paper Award, FSDM 2016 Best Oral Presentation, ICASSP 2019 and LVA/ICA 2010 Best Student Paper Award Nominees. He is a Senior Area Editor (2019-2023) for IEEE Transactions on Signal Processing, an Associate Editor (2020-) for IEEE/ACM Transactions on Audio Speech and Language Processing, an Associate Editor of Nature Scientific Report (2022-) and Senior Area Editor of Digital Signal Processing (2021-), and an Associate Editor (2019-) for EURASIP Journal on Audio Speech and Music Processing. He is a Specialty Editor in Chief (2021-) of Frontier in Signal Processing, and was an Associate Editor (2014-2018) for IEEE Transactions on Signal Processing. He is elected Chair (2023-2024) of IEEE SPS Machine Learning for Signal Processing Technical Committee, elected Vice Chair (2022-2024) of EURASIP Technical Area Committee on Acoustic Speech and Music Signal Processing, an elected Member (2021-2023) of the IEEE Signal Processing Theory and Methods Technical Committee, and an elected Member (2019-) of the International Steering Committee of Latent Variable Analysis and Signal Separation. He was a Publication Co-Chair for ICASSP 2019 (Brighton, UK) and a Satellite Workshop Co-Chair for INTERSPEECH 2022 (Incheon, Korea), and a Technical/Program Committee Member of over 100 international conferences.

Pei Xiao is a professor of Wireless Communications at the Institute for Communication Systems (ICS), home of 5GIC and 6GIC at the University of Surrey. He received the PhD degree from Chalmers University of Technology, Gothenburg, Sweden in 2004. He is currently the technical manager of 5GIC/6GIC, leading the research team in the new physical layer work area, and coordinating/supervising research activities across all the work areas (https://www.surrey.ac.uk/institute-communication-systems/5g-6g-innovation-centre). Prior to this, he worked at Newcastle University and Queen's University Belfast. He also held positions at Nokia Networks in Finland. He has published extensively in the fields of communication theory, RF and antenna design, signal processing for wireless communications, and is an inventor on over 15 recent 5GIC patents addressing bottleneck problems in 5G systems.