Deep Learning Based Joint Resource Scheduling Algorithms for Hybrid MEC Networks

Feibo Jiang, Kezhi Wang, Li Dong, Cunhua Pan, Wei Xu and Kun Yang

***Abstract*—In this paper, we consider a hybrid mobile edge computing (H-MEC) platform, which includes ground stations (GSs), ground vehicles (GVs) and unmanned aerial vehicle (UAVs), all with mobile edge cloud installed to enable user equipments (UEs) or Internet of thing (IoT) devices with intensive computing tasks to offload. Our objective is to obtain an online offloading algorithm to minimize the energy consumption of all the UEs, by jointly optimizing the positions of GVs and UAVs, user association and resource allocation in real-time, while considering the dynamic environment. To this end, we propose a hybrid deep learning based online offloading (H2O) framework where a large-scale path-loss fuzzy c-means (LSFCM) algorithm is first proposed and used to predict the optimal positions of GVs and UAVs. Secondly, a fuzzy membership matrix U-based particle swarm optimization (U-PSO) algorithm is applied to solve the mixed integer nonlinear programming (MINLP) problems and generate the sample datasets for the deep neural network (DNN) where the fuzzy membership matrix can capture the small-scale fading effects and the information of mutual interference. Thirdly, a DNN with the scheduling layer is introduced to provide user association and computing resource allocation under the practical latency requirement of the tasks and limited available computing resource of H-MEC. In addition, different from traditional DNN predictor, we only input one UE's information to the DNN at one time, which will be suitable for the scenarios where the number of UE is varying and avoid the curse of dimensionality in DNN.**

***Index Terms*—Mobile edge computing, resource allocation, UAV, fuzzy c-means, particle swarm optimization, deep neural network.**

I. Introduction

Computation intensive applications, such as face recognition, language processing, online gaming, augmented reality (AR) and virtual reality (VR), have been fast developing and increasingly outgrowing the limited capabilities of devices [1]. Thanks to the recent advancement of mobile edge computing (MEC) technology, the gap between the limited amount of resource in devices and the demands for better experience is being reduced [2], [3]. With the help of MEC, the UE can offload the intensive computations to the nearby edge servers to save energy consumption and increase the computational capacity [4], [5]. However, different from the traditional cloud in data center, edge cloud may be implemented by the router, switches, which may have some free computing resource and are closer to the users. Recently, vehicle and UAV [6] based MEC has also been proposed. Due to limited amount of the computing resource in MEC and finite physical bandwidth of wireless channels, task admission control and resource allocation are normally required, especially in the presence of a large number of delay-sensitive tasks. The problem is generally formulated as mixed integer nonlinear programming (MINLP). To tackle the MINLP problem, branch-and-bound algorithms [7] and dynamic programming [8] are normally used to obtain the globally optimal offloading solution. However, the search spaces of both methods increase exponentially with the network size and are computationally prohibitive for large-scale MEC networks. To reduce the computational complexity, heuristic local searching methods are proposed. For instance, [9] proposed a coordinate descent (CD) method that searches along one binary variable at a time. A similar heuristic search method for multi-server MEC networks was studied in [10], which iteratively adjusts binary offloading decisions. Another widely adopted heuristic method is through convex relaxation, e.g., by relaxing integer variables to be continuous between 0 and 1 [11] or by approximating the binary constraints with quadratic constraints [12].

Nonetheless, on one hand, the solution quality of reduced-complexity heuristics is not guaranteed. On the other hand, both searching-based and convex relaxation methods often require a large amount of iterations for an algorithm to reach a satisfying local optimum. Hence, they are unsuitable for real-time processing in fast changing environment, as the optimization problem needs to be re-solved once the number and position of UEs have varied significantly.

Recently, some artificial intelligence methods have emerged as effective tools for enhancing MECs. In [13], the energy-efficient computation offloading management scheme in the MEC system with small cell networks (SCNs) is proposed, and a hierarchical genetic algorithm (GA) and particle swarm optimization (PSO)-based heuristic algorithm are designed to solve this problem. In [14], a deep learning (DL) algorithm

This work was supported in part by the National Natural Science Foundation of China under Grant no. 41604117, 41904127, 41874148, 61620106011, 61572389 and 61871109. This work was also supported in part by the Royal Academy of Engineering under the Distinguished Visiting Fellowship scheme (DVFS21819\9\7) and by Scientific Research Fund of Hunan Provincial Education Department in China under Grant no. 18A031

Feibo Jiang (jiangfb@hunnu.edu.cn) is with Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, Changsha, China, Kezhi Wang (kezhi.wang@northumbria.ac.uk) is with the department of Computer and Information Sciences, Northumbria University, UK, Li Dong (Dlj2017@hunnu.edu.cn) is with Key Laboratory of Hunan Province for New Retail Virtual Reality Technology, Hunan University of Technology and Business, Changsha, China, Cunhua Pan (Email: c.pan@qmul.ac.uk) is with School of Electronic Engineering and Computer Science, Queen Mary University of London, London, E1 4NS, UK, Wei Xu (wxu@seu.edu.cn) is with NCRL, Southeast University, Nanjing, China, Kun Yang (kunyang@essex.ac.uk) is with the School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun, China and also with the School of Computer Sciences and Electrical Engineering, University of Essex, CO4 3SQ, Colchester, UK.

Corresponding authors: Kezhi Wang and Kun Yang

based on multi-long and short-term memory (LSTM) networks is proposed to forecast the traffic of small base stations (SBSs), on the basis of which an offline mobile data offloading strategy obtained through on cross-entropy is presented. In [15], a conceptor-based echo state network is proposed to predict content request distribution of users and its mobility pattern when the network is available. Based on the prediction results, the optimal positions of UAVs and the content to cache at UAVs can be obtained. In [16], a distributed deep learning offloading algorithm is introduced in MEC networks, where multiple parallel DNNs are trained separately and applied to make offloading decisions cooperatively. In [17], an emerging deep neural network technique is used in the mobile crowd sensing (MCS). The proposed technique employs convolutional neural network for feature extraction, and then directs the sensing and movement of UEs under the guidance of the distributed multi-agent deep deterministic policy gradient method. In [18], a deployment strategy for the distributed multi-layer convolutional neural network is presented. The strategy divides the convolutional neural network into two parts: the preprocessing part and the classification part. The preprocessing part is deployed on the edge server for feature extraction and compression of the image data so as to reduce the data transmission between the edge server and the cloud server.

Nonetheless, on one hand, the heuristic computation algorithms have excellent global search ability and high calculation accuracy, but they need long computing time. On the other hand, the supervised learning algorithms have outstanding prediction and reasoning capabilities, but they require a large amount of labelled training data.

In this paper, we consider a hybrid mobile edge computing (H-MEC) platform, where there are ground stations (GSs), ground vehicles (GVs) and unmanned aerial vehicle (UAVs), all with edge cloud enhanced, which can enable UEs with computational intensive tasks to offload. We aim to obtain an online offloading algorithm to minimize the energy consumption of all the UEs, by jointly optimizing the positions of GVs and UAVs, user association and resource allocation in real time, while considering the dynamic environment. Towards this end, we propose a hybrid deep learning based online offloading (H2O) framework to achieve the above targets. Compared with the existing integer programming and learning based methods, we have the following novel contributions:

- We first introduce a large-scale path-loss fuzzy c-means (LS-FCM) clustering algorithm to locate the positions of UAVs and GVs, which has two improvements compared to the traditional FCM: First, it can fix some cluster centers denoted as GS positions and not allow them to participate in the iteration process. Second, it introduces the large-scale path-loss component to replace distance in clustering process.
- We then introduce a fuzzy membership matrix U-based particle swarm optimization (U-PSO) algorithm, which can solve the task admission and resource allocation problem with different initial states. This procedure is repeated until enough samples are collected. PSO can solve the complex MINLP problems precisely and provide high quality labeled samples to the DNN for offline training. Additionally, a U-based roulette wheel selection strategy is applied to guide the initial stage of PSO and provide high quality initial solution for accelerating convergence of PSO, where the fuzzy membership matrix can capture the small-scale fading and the information of mutual interference.
- The DNN is applied for real-time decision-making. The training stage is done by the collected samples from U-PSO, by applying the continuous membership information as the input for offloading action generation and resource allocation. This hybrid mechanism keeps the advantages of the PSO in finding global optimal solutions, while speeding up the decision-making through DNN. Besides, to generate an offloading action, the proposed H2O framework only needs to input the membership information of one UE each time. Compared to some conventional deep learning methods that require to input the information of all UEs, H2O is computationally feasible and efficient in large-size networks with different number of UEs, which is suitable for continually dynamic scenarios. Moreover, the online new input and output of the DNN will be collected, recalculated and stored back to the sample memory database, which is very important for improving the performance of DNN in real scenarios.
- We further develop an additional scheduling layer of DNN to check if the constraints are guaranteed. Also, admission control is conducted in this layer. The H2O framework is suitable for solving large-scale MINLP problem in real-time. We finally evaluate the proposed H2O framework under extensive numerical studies. The experiment results of the proposed H2O are compared with several different kinds of benchmark solutions, which demonstrates the feasibility and effectiveness of our framework. The simulation results have also shown that our solutions have better computational efficiency and accuracy. This can make real-time and optimal design feasible in the H-MEC networks even in a fast changing environment.

The remainder of this paper is organized as follows. In Section II, we describe the system model and problem formulation. We introduce the detailed designs of the H2O algorithm in Section III. Numerical results are presented in Section IV. Finally, the paper is concluded in Section V.

II. System model and problem formulation

Fig. 1 shows our proposed H-MEC networks with several GSs, GVs and UAVs, all of which have edge server enhanced. The locations of GSs are assumed to be fixed whereas the locations of GVs and UAVs can be optimized.

We consider that there are N UEs randomly distributed in the ground, each of which has a computation task to be executed. The set of UEs is denoted as $\mathcal{N} = \{1, 2, \cdots, N\}$. Consider there are J UAVs, K GVs and M GSs, which can enable the UEs to offload the tasks. The sets of UAVs, GVs and GSs are denoted as $\mathcal{J} = \{1, 2, \cdots, J\}$, $\mathcal{K} = \{1, 2, \cdots, K\}$ and $\mathcal{M} = \{1, 2, \cdots, M\}$, respectively. We use variables a_i^L,

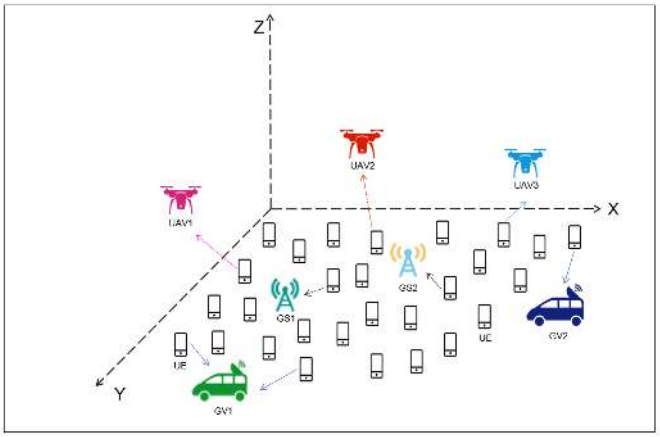

Fig. 1: The H-MEC network.

$a_{i,j}^{MEC}$ to denote the possible places where the UEs can execute the tasks, where a_i^L and $a_{i,j}^{MEC}$ denote the local executing, and offloading to the H-MECs (including UAVs, GVs and GSs), respectively. Then we have the following:

$$a_i^L = \{0,1\}, \forall i \in \mathcal{N}$$
$$a_{ij}^{MEC} = \{0,1\}, \forall i \in \mathcal{N}, \forall j \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\} \quad (1)$$

where $a_i^L = 1$ means that the i-th UE decides to execute the task itself, and $a_i^L = 0$ otherwise, $a_{ij}^{MEC} = 1$ means that the i-th UE decides to offload the task to the j-th UAV (when $j \in \mathcal{J}$), or to the j-th GV (when $j \in \mathcal{K}$), or to the j-th GS (when $j \in \mathcal{M}$), and $a_{ij}^{MEC} = 0$ otherwise. It is assumed that each task can be executed at most one place. Then, we have

$$a_i^L + \Sigma_{j\in\{\mathcal{J},\mathcal{K},\mathcal{M}\}}\ a_{ij}^{MEC} = 1, \forall i \in \mathcal{N}. \quad (2)$$

Similar to [19], we assume that the i-th UE has the computational intensive task Q_i to be executed as follows:

$$Q_i = (F_i, D_i, T^{req}), \forall i \in \mathcal{N} \quad (3)$$

where F_i describes the total number of the CPU cycles of Q_i to be computed, D_i denotes the data size transmitting to the H-MECs if the offloading action is decided and T^{req} is the latency constraint or quality of service (QoS) requirement of this task. Without loss of generality, in this paper, we consider that all the tasks have the same time requirement T^{req}. D_i and F_i can be obtained by using the approaches provided in [20]. If the task is selected to offload, the execution time of the task is given by:

$$T_{ij}^C = \frac{F_i}{f_{ij}} \quad (4)$$

where f_{ij} is the computation capacity of the j-th H-MEC providing to the i-th UE. Also, the time to offload the data is given by:

$$T_{ij}^{Tr} = \frac{D_i}{r_{ij}} \quad (5)$$

where r_{ij} is the offloading data rate from the i-th UE to the j-th place. Then, we can have:

$$T_{ij}^{Tr} + T_{ij}^C = \frac{D_i}{r_{ij}} + \frac{F_i}{f_{ij}} \leq T^{req} \quad (6)$$

which means that each task must meet the latency requirement. If this task is executed in UE itself, one has

$$\frac{F_i}{f_i^L} \leq T^{req} \quad (7)$$

where f_i^L is the local executing capacity. Therefore, one can have

$$a_i^L \frac{F_i}{f_i^L} + \sum_{j\in\{\mathcal{J},\mathcal{K},\mathcal{M}\}} a_{ij}^{MEC}\left(\frac{D_i}{r_{ij}} + \frac{F_i}{f_{ij}}\right) \leq T^{req}, \forall i \in \mathcal{N}. \quad (8)$$

Also, assume that the computing capacities in the UEs, UAVs, GVs and GSs are limited as

$$a_i^L f_i^L \leq F_{i,max}^L, \forall i \in \mathcal{N}$$
$$\sum_{i\in\mathcal{N}} a_{ij}^{MEC} f_{ij} \leq F_{j,max}^{MEC}, \forall j \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\} \quad (9)$$

where $F_{i,max}^L$ is the local computational capability of the i-th UE, $F_{j,max}^{MEC}$ is the remote computational capability of the j-th H-MEC. The power consumption in the i-th UE can be given by

$$P_i^{ue} = \begin{cases} \sum_{j\in\{\mathcal{J},\mathcal{K},\mathcal{M}\}} a_{ij}^{MEC} P_{ij}^T, & \text{if offloading} \\ P_i^E, & \text{if local execution} \end{cases} \quad (10)$$

where P_{ij}^T is the transmitting power from the i-th UE to the j-th H-MEC and P_i^E is the execution power in the i-th UE if UE conducts the task itself and

$$P_i^E = k_i (f_i^L)^{v_i}, \forall i \in \mathcal{N} \quad (11)$$

where $k_i \geq 0$ is the effective switched capacitance and $v_i \geq 1$ is the positive constant. To match the realistic measurements, we set $k_i = 10^{-27}$ and $v_i = 3$ [21].

Assume that the coordinates of the i-th UE, the j-th UAV, the j-th GV and the j-th GS are (x_i, y_i), $(X_j^{UAV}, Y_j^{UAV}, H_j^{UAV})$, (X_j^{GV}, Y_j^{GV}), and (X_j^{GS}, Y_j^{GS}), respectively. Then, the horizontal distance between the i-th UE and the j-th UAV is given by

$$R_{ij}^{UAV} = \sqrt{(X_j^{UAV} - x_i)^2 + (Y_j^{UAV} - y_i)^2}, \forall i \in \mathcal{N}, \forall j \in \mathcal{J}. \quad (12)$$

If UEs decide to offload the task to UAVs, the data rate can be given as

$$r_{ij} = B\log_2\left(1 + \frac{P_{ij}^T h_{ij}^{UAV}}{\sigma^2}\right), \forall i \in \mathcal{N}, \forall j \in \mathcal{J} \quad (13)$$

where B is the channel bandwidth, h_{ij}^{UAV} is the channel quality information (CQI), denoted as $h_{ij}^{UAV} = \alpha_j^{UAV}\left(\sqrt{H_j^{UAV^2} + R_{ij}^{UAV^2}}\right)^{-2}$, α_j^{UAV} is the small-scale fading component, $\left(\sqrt{H_j^{UAV^2} + R_{ij}^{UAV^2}}\right)^{-2}$ is the large-scale path-loss component [22]. Note that we assume that the users offload their tasks to UAV via orthogonal frequency division multiplexing (OFDM) channels, which means that there is no interference between each other. Similar to [23], we assume that horizontal coverage of UAV is constrained by the following

$$R_{ij}^{UAV} \leq H_j^{UAV} \tan\phi_j^{UAV}, \forall i \in \mathcal{N}, \forall j \in \mathcal{J} \quad (14)$$

where ϕ_j can be decided by the angle of antenna in the UAV [23].

Then, the horizontal distance between the i-th UE and the j-th GV is

$$R_{ij}^{GV} = \sqrt{(X_j^{GV} - x_i)^2 + (Y_j^{GV} - y_i)^2}, \forall i \in \mathcal{N}, \forall j \in \mathcal{K}. \tag{15}$$

If UEs decide to offload to the GVs, the data rate can be given as:

$$r_{ij} = B\log_2\left(1 + \frac{P_{ij}^T h_{ij}^{GV}}{\sum_{k\in\mathcal{N}k\neq i} P_{kj}^T h_{kj}^{GV} + \sigma^2}\right), \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{K} \tag{16}$$

where we assume the UEs share the same channel when they decide to offload to the same GV and there is interference between them. $h_{ij}^{GV} = \alpha_j^{GV}\left(R_{ij}^{GV}\right)^{-2}$, α_j^{GV} is the small-scale fading component and $\left(R_{ij}^{GV}\right)^{-2}$ is the large-scale path-loss component.

The horizontal distance between the i-th UE and the j-th GS is as

$$R_{ij}^{GS} = \sqrt{(X_j^{GS} - x_i)^2 + (Y_j^{GS} - y_i)^2}, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}. \tag{17}$$

If UEs decide to offload to the GS, the data rate can be given as

$$r_{ij} = B\log_2\left(1 + \frac{P_{ij}^T h_{ij}^{GS}}{\sum_{k\in\mathcal{N}k\neq i} P_{kj}^T h_{kj}^{GS} + \sigma^2}\right), \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \tag{18}$$

where $h_{ij}^{GS} = \alpha_j^{GS}\left(R_{ij}^{GS}\right)^{-2}$, α_j^{GS} is the small-scale fading component and $\left(R_{ij}^{GS}\right)^{-2}$ is the large-scale path-loss component.

Define the decision variables as $\mathbf{A} = \{a_i^L, a_{ij}^{MEC}\}$, $\forall i \in \mathcal{N}$, $\forall j \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\}$, $\mathbf{F} = \{f_i, f_{ij}\}$, $\forall i \in \mathcal{N}$, $\forall j \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\}$, the location for UAV as $\mathbf{W} = \{X_j^{UAV}, Y_j^{UAV}, H_j^{UAV}\}$, $\forall j \in \mathcal{J}$, and the location for GV as $\mathbf{G} = \{X_j^{GV}, Y_j^{GV}\}$, $\forall j \in \mathcal{K}$. Then, one can formulate the energy minimization optimization as

$$P1: \min_{\mathbf{A},\mathbf{F},\mathbf{W},\mathbf{G}} \sum_{i\in\mathcal{N}} \sum_{j\in\{\mathcal{J},\mathcal{K},\mathcal{M}\}} a_{ij}^{MEC} P_{ij}^T \frac{D_i}{r_{ij}} + \sum_{i\in\mathcal{N}} a_i^L P_i^E \frac{F_i}{f_i^L}$$
$$s.t. \qquad (1), (2), (8), (9), (14). \tag{19}$$

One can see that Problem $(P1)$ is an MINLP problem, which is hard to solve in general. This is because the admission decision $\mathbf{A}$ is binary while the resource allocation decision $\mathbf{F}$ and locations of UAV $\mathbf{W}$ and GV $\mathbf{G}$ are continuous. The major difficulty of solving $P1$ lies in three aspects: (1) large-scale mixed integer programming optimization, (2) real-time decision-making and (3) the dynamic environment. To circumvent these hurdles, we propose a novel H2O framework by first applying FCM clustering algorithm to get the positions of GVs and UAVs. Then, DNN is trained offline by using the samples obtained from PSO with the global search. After that, DNN can be applied to make the real time decision, based on the input of fuzzy membership information, even in a fast changing environment. The important notations used throughout this paper are summarized in TABLE I.

TABLE I: Notations used throughout this paper.

Notation	Description
$\mathcal{N}$	The set of UEs
$\mathcal{J}$, $\mathcal{K}$, $\mathcal{M}$	The sets of UAVs, GVs and GSs
a_i^L	Local executing indicator
$a_{i,j}^{MEC}$	Offloading indicator
P_{ij}^T	Transmitting power
P_i^E	Local execution power
D_i	The transmitting data size
F_i	The required number of the CPU cycles
f_i^L	Local executing capacity
r_{ij}	Data rate
h_{ij}	Channel quality information
T^{req}	The latency constraint
f_{ij}	Computation capacity of the j-th H-MEC providing to the i-th UE
α_j	The small-scale fading component
R_{ij}^{-2}	The large-scale path-loss component
$\mathbf{C}$	Cluster center matrix
τ	Weighting exponent of FCM
c	The number of clusters
ε	Convergence threshold of FCM
T_{FCM}	The maximum iteration number of FCM
$\mathbf{U}$	Fuzzy membership matrix
$\boldsymbol{x}_i$	Position vector of PSO
$\boldsymbol{v}_i$	Velocity vector of PSO
w_{max}	The initial inertia weight of PSO
w_{min}	The final inertia weight of PSO
$\boldsymbol{p}_g$	The global best particle
P	The number of particles
T_{PSO}	The maximum iteration number of PSO
$\boldsymbol{\theta}$	Network parameters of DNN
β	The learning rate of DNN
$\mathbf{r}_\ell$	The ℓ-th layer output of DNN
$\boldsymbol{u}_i$	Fuzzy membership information of the i-th UE
N_c	The number of constraints
T_{CNN}	The maximum iteration number of CNN

III. The hybrid deep learning-based online offloading framework

A. Algorithm Overview

In this section, we provide a novel algorithm named hybrid deep learning based online offloading (H2O) to solve Problem $P1$. The structure of the H2O algorithm is illustrated in Fig. 2, which is composed of four parts: GV and UAV location optimization, sample collection, DNN offline learning and DNN online decision. The main procedure of the H2O algorithm is divided into the offline training phase and the online optimization phase.

The offline training phase is carried out in the remote cloud server with high computational and storage capabilities. We regard the optimization problem $P1$ as the mapping function from the input system parameters to the output resource allocation solutions. We use a DDN to fit this mapping function. To this end, we need to find the resource allocation solution under certain system parameters. In other words, we need samples to train the DNN. To this end, we provide one algorithm to solve Problem $P1$ to obtain the training samples. In specific, we first adopt the LS-FCM algorithm in Subsection-B to obtain the locations of the UAVs and GVs. We then deploy the UAVs and GVs according to the calculated locations. Then, we propose a novel U-PSO algorithm in Subsection-C to obtain the offloading selection and resource allocation solution based on the novel fuzzy membership information that can capture the small-scale channel information and the relative interference among the UEs. Finally, supervised learning algorithm is used to train the DNN that can be applied for the scenarios where the number of UEs is varying.

Then, the trained DNN can be implemented for online calculation. In particular, each UE only needs to input its membership values (detailed in Subsection-C), then the DNN can output its offloading choice and computing resource allocation result. This has much lower computational complexity since it only needs to perform some simple algebraic calculations instead of solving the original optimization problem through high-complexity heuristic algorithms. In addition, during the online implementation, some results output by the DNN will be regarded as new samples and be stored in the database at the cloud. These samples will be used for offline training and to update the DNN. This is very important since it can track the variations of the real scenarios.

In the following, we provide more details of each step of the H2O algorithm.

B. Location optimization based on LS-FCM

Clustering algorithm is always applied to determine the locations of UAVs [15], but it cannot be directly used in our work. First, the positions of GSs are fixed, but the positions of GVs and UAVs are varying. The conventional clustering method did not consider the case when some points are fixed. Second, the iterative process of conventional FCM only considers the distances between UEs and MECs, and does not consider CQI between them. To solve these problems, we propose a novel large-scale path-loss fuzzy c-means (LS-FCM) algorithm to locate the positions of UAVs and GVs, which has two advantages: First, it can fix some cluster centers denoted as GS positions and not allow them to participate in the iteration process. Second, it introduces the large-scale path-loss component to replace distance in clustering process.

FCM clustering is a data clustering method in which each data point belongs to a fuzzy cluster, with a degree specified by a membership grade. FCM partitions a collection of n data point $\boldsymbol{z}_i$ into fuzzy clusters. In our study, $\boldsymbol{z}_i$ is the i-th data point which denotes the position (x_i, y_i) of the i-th UE.

Specifically, the objective function of LS-FCM is expressed in the following:

$$G = \sum_{i \in \mathcal{N}} \sum_{j \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\}} (\mu_{ij})^{\tau} d'_{ij} \tag{20}$$

where $d'_{ij} = 1/R_{ij}^{-2}$ denotes the large-scale path-loss component, μ_{ij} denotes the fuzzy relationship between the i-th UE and the j-th MEC. Here, $\boldsymbol{C}_j$ is denoted as the j-th cluster center which denotes the position of the j-th H-MEC (including UAVs, GVs and GSs). $\tau(\tau > 1)$ is a scalar termed as the weighting exponent that controls the fuzziness of the resulting clusters. To minimize the criterion G, the centroid $\boldsymbol{C}_j$ is updated according to Eq. (21), and μ_{ij} is updated according to Eq. (22), respectively,

$$\boldsymbol{C}_j = \frac{\sum_{i \in \mathcal{N}} (\mu_{ij})^{\tau} z_i}{\sum_{i \in \mathcal{N}} (\mu_{ij})^{\tau}}, \forall j \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\}, \tag{21}$$

and

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d'_{ij}}{d'_{ik}} \right)^{\frac{2}{\tau - 1}}}, \tag{22}$$

where c is the number of clusters.

μ_{ij} and $\boldsymbol{C}_j$ are calculated iteratively through these two equations until the FCM algorithm converges. Based on the solutions of $\mathbf{C} = [\boldsymbol{C}_j]$, we can decide the locations of UAVs and GVs. The reason why we use FCM to locate the position of UAVs and GVs is that the LS-FCM clustering algorithm aims at minimizing the total large-scale path loss of all UEs as seen in Eq. (20), which implicitly reduces the system energy shown in the objective function of Problem (19).

Finally, a hard classification is adopted by assigning each UE solely to the cluster with the highest μ_{ij}. In the LS-FCM, since the GS positions are fixed, some cluster centers are set to the GS positions in advance and are not allowed to participate in the iteration process. According to the hard classification, the cluster centers are sorted in a descending order according to the total required computing resources of all UEs in this cluster. Then the centers of the cluster are assigned to GVs and UAVs based on their available computing resources. For example, the first center of the cluster is assigned to the H-MEC with the most computing resources, the last center of the cluster is assigned to the H-MEC with the least computing resources. Please note that this is not the final computing offloading solution. This stage just provides a rough estimation of computing resource needed by all UEs in each cluster. This LS-FCM just provides the locations for each H-MEC. Although, some UEs are in one H-MEC's cluster, these UEs may not be offloaded to this H-MEC since the small-scale fading are not considered in LS-FCM. The offloading

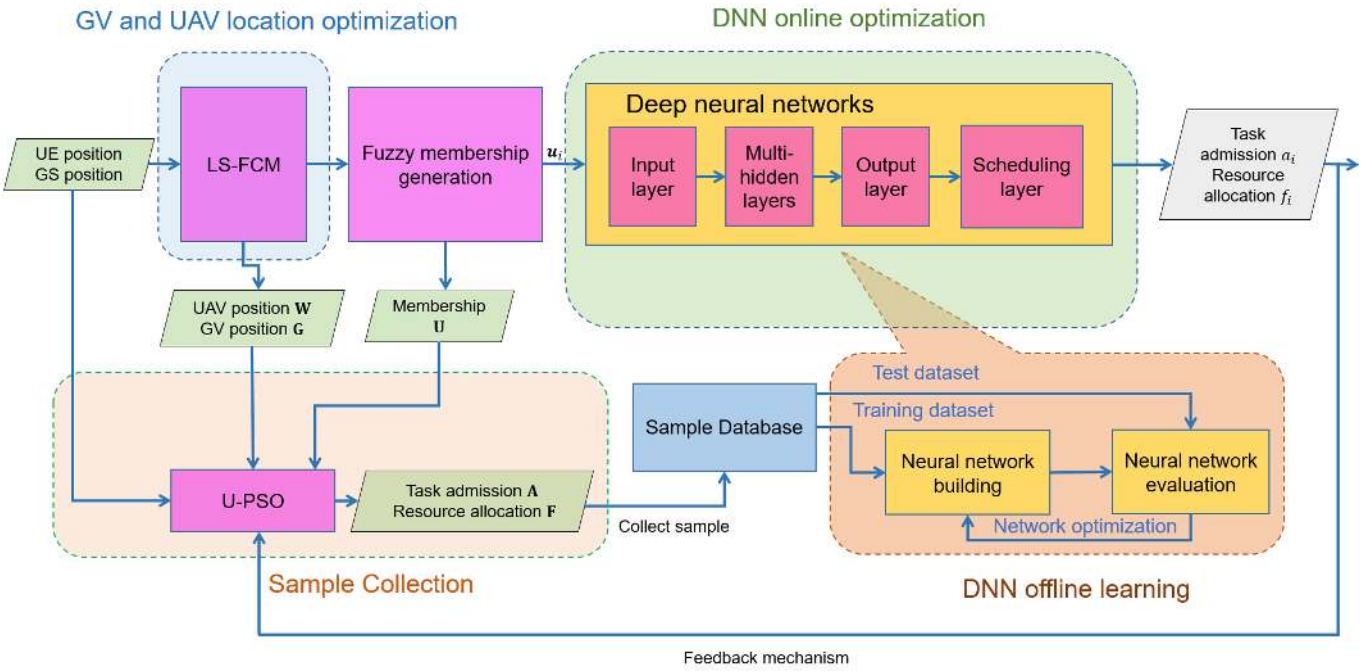

Fig. 2: The proposed H2O framework.

selection and computing allocation will be studied in the next subsection.

Several stopping rules can be used in LS-FCM. One is to terminate the algorithm when the maximum iteration number T_{FCM} is reached or when the variation of objective function ΔG is less than threshold ε.

The detailed description of LS-FCM algorithm is provided in **Algorithm 1**.

Algorithm 1 LS-FCM algorithm

Input: GS positions, UE positions, ε, c, τ, T_{FCM}.
Output: $\mathbf{C}$.

1: Initialize the locations of cluster centers. Calculate μ_{ij} according to Eq. (22).
2: Fix some centers of $\mathbf{C}$ according to the GS positions.
3: $\Delta G(1) = 1, G(0) = 0, t = 1$.
4: **while** $|\Delta G(t)| > \varepsilon$ and $t < T_{FCM}$ **do**
5: Calculate the remaining cluster centers of matrix $\mathbf{C}$ using Eq. (21).
6: Calculate the objective function $G(t)$ using Eq. (20).
7: Update each μ_{ij} using Eq. (22).
8: $\Delta G(t) = G(t) - G(t-1)$.
9: $t = t + 1$.
10: **end while**

C. Computing offloading selection and computing resource allocation based on U-PSO

When the locations of UAV and GV are determined by the LS-FCM algorithm, we need to optimize the computing offloading decision for each UE and the corresponding allocated computing resources based on both the large-scale path-loss and small-scale channel fading information. This topic will be studied in this subsection. Heuristic algorithms are always used to solve the complex MINLP problem and obtain high-quality global solution [13]. However, heuristic algorithms also exhibit some issues: it takes more calculation time than traditional gradient descent method when you want to achieve similar accuracy and the convergence speed decreases considerably in the later period of evolution. So the heuristic algorithm is fundamentally infeasible for real-time system optimization under fast changing environment. In this paper, we propose a U-PSO, in which we use PSO algorithm to generate samples for DNN offline training and then use trained DNN to make online task admission and resource allocation decision.

PSO is a heuristic computation technique that was originally proposed by Eberhart and Kennedy [24]. The standard PSO can be described as follows. Consider a swarm with P particles, each particle is a candidate solution and it moves in a bounded p-dimensional search space. The position vector and velocity vector of the i-th particle are represented as $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$ and $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \cdots, v_{ip})$, respectively. PSO explores the search space by modifying the velocity of each particle, according to two attractors: the best position found so far by the particle itself $\boldsymbol{p}_{bi}$, and the best position identified so far by the whole swarm $\boldsymbol{p}_g$. By integrating these two attractors, the particle behavior can be modeled by using the following equations:

$$\begin{aligned} v_{id}(t+1) =& w(t)v_{id}(t) + c_{cog} \cdot rand_1(p_{bid}(t) - x_{id}(t)) \\ &+ c_{soc} \cdot rand_2(p_{gd}(t) - x_{id}(t)) \end{aligned} \tag{23}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{24}$$

where c_{cog} is the cognitive factor and c_{soc} is the social factor; $rand_1$ and $rand_2$ are generated randomly between 0 and 1; $p_{bid}(t)$ is the best position of the i-th particle at iteration t;

$p_{gd}(t)$ is the best particle position among all the particles at iteration t; d=1,2,...,p. $w(t)$ is the inertia weight. Typically, the inertia weight is set according to [25]:

$$w(t) = w_{max} - (w_{max} - w_{min}) \cdot t/T_{pso} \tag{25}$$

where w_{max} is the initial inertia weight, w_{min} is the final inertia weight, T_{pso} is the maximum iteration number.

The energy-efficient admission of delay-sensitive tasks is a complex MINLP problem, using traditional PSO algorithm to solve this problem has three drawbacks that avoids its direct application in our considered optimization problem. Firstly, PSO algorithm often employs continuous real-valued encodings, but the admission decision matrix $\mathbf{A}$ is a matrix with integer elements equal to 0 or 1; Second, traditional PSO did not need to check the constraints; Third, traditional PSO initializes the particle population randomly, and it does not take advantage of the CQI information. In this regard, we propose a new U-based particle swarm optimization (U-PSO) algorithm to solve the MINLP problem efficiently.

Firstly, we improve the coding of particle. In our U-PSO algorithm, the particle can be represented as:

$$\begin{aligned}\boldsymbol{x} =&[a_1, a_2, \cdots, a_i, \cdots, a_N, f_1, f_2, \cdots, f_i, \cdots, f_N] \\ =&[\boldsymbol{x}_a, \boldsymbol{x}_f]\end{aligned} \tag{26}$$

where $a_i = 0$ means that the i-th UE decides to execute the task itself, and $a_i = k$ means that the i-th UE decides to offload the task to the k-th H-MEC, while $k \in \{1, 2, \cdots, J + K + M\}$. $f_i \in \mathbb{R}$ means the allocated resource to the i-th UE. If $a_i = 0$, $f_i = f_i^L$. This representation transforms the decision matrix $\mathbf{A}$ and resource allocation matrix $\mathbf{F}$ to a terse coding for PSO. Then we can round $\boldsymbol{x}_a$ part of the particle after each iteration.

Secondly, we add a constraint check step evaluating each particle. If a particle leads to constraint violation, we will assign a large penalty ρ to the fitness value which is set to be equal to the objective function of (19).

Thirdly, the final solution of the U-PSO algorithm mainly depends on the initial input of the algorithm. Improper selection of the initial point may cause the final solution to be stuck at a locally optimal point with low performance. Hence, it is critical to select a good initial point. In this paper, we use U-based roulette wheel selection strategy to provide high-quality initial solution for accelerating convergence. However, to implement the U-based roulette wheel, we need to know the probability of each UE to select which H-MEC to offload. One intuitive method is to use the parameter μ_{ij} in (22) after the LS-FCM algorithm in the above subsection. In specific, the probability of the i-th UE to select the j-th H-MEC is given by

$$p_{ij} = \frac{\mu_{ij}}{\sum\limits_{k \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\}} \mu_{ik}}. \tag{27}$$

However, this selection does not capture the instantaneous small-scale channel information and interference among the UEs. To resolve this issue, we provide a novel fuzzy membership matrix based method. It applies the small-scale fading and interference information to generate dynamic membership information, so it can output a novel fuzzy membership matrix, which includes the CQI and interference information for UEs. The fuzzy membership matrix is very important in our framework. It can not only be used here for generating the initial point of the U-PSO algorithm, but also serves as the input as the DNN discussed in the next subsection.

we define the novel fuzzy membership matrix $\mathbf{U} = [u_{ij}]$ including the small-scale fading and interference information, then we update the fuzzy membership matrix in the dynamically changing environment. The membership degree u_{ij} is expressed as following:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{h'_{ij}}{h'_{ik}}\right)^{\frac{2}{\tau-1}}}, \tag{28}$$

where $h'_{ij} = \left(\gamma \sum_{k \in \mathcal{N}, k \neq i} P_{kj}^T \alpha_j R_{kj}^{-2}\right) / P_{ij}^T \alpha_j R_{ij}^{-2}$ is an enhanced version of d'_{ij}, which includes the small-scale fading component α_j for the j-th MEC and the referenced interference information $\sum_{k \in \mathcal{N}, k \neq i} P_{kj}^T \alpha_j R_{kj}^{-2}$, $\forall j \in \{\mathcal{K}, \mathcal{M}\}$, and γ is the trade-off factor. The fuzzy membership matrix $\mathbf{U}$ can reflect the changes of channel and the interferences of environment in real time.

By using the updated dynamic membership information defined in Eq. (28), we now can obtain the probability of UE i to select H-MEC j is given by

$$p_{ij} = \frac{u_{ij}}{\sum\limits_{k \in \{\mathcal{J}, \mathcal{K}, \mathcal{M}\}} u_{ik}}. \tag{29}$$

With the probability values, we can now employ the U-based roulette wheel to initialize the initial solution. For example, suppose there are five H-MECs that UE i can choose to offload (three UAVs, one GV and one GS), in Fig. 3 the circumference of the roulette wheel is equal to one. UAV1 is the most fit individual and has the largest probability to be selected, whereas GS and GV are the least fit and have smaller intervals within the roulette wheel. To select a H-MEC, a random number is generated in the interval [0,1], and the H-MEC whose segment spans the random number is selected and UE i will offload its task to this H-MEC. This process is repeated until all UEs have selected their H-MECs. Then based on the offloading decision, the resource of each H-MEC is allocated evenly to its associated UEs. The process of U-PSO algorithm is present in **Algorithm 2**.

The fuzzy membership matrix $\mathbf{U}$ which captures small-scale fading and mutual interference information plays a key role in our H2O framework. The fuzzy membership matrix $\mathbf{U}$ can not only guide the initialization of U-PSO, but also provide a concise representation of the relationship between UEs and H-MECs, which will be a perfect input of DNNs.

D. Offline DNN training and online implementation

Given the locations of UAVs and GVs, the Problem $P1$ can be regarded as an unknown function mapping from the fuzzy membership matrix $\mathbf{U}$ to the optimal task admission $\mathbf{A} = [a_1, a_2, \cdots, a_i, \cdots, a_N]$ and resource allocation $\mathbf{F} = [f_1, f_2, \cdots, f_i, \cdots, f_N]$, namely.

$$\mathcal{F}\colon \mathbf{U} \in \mathbb{R}^{N \times (J+K+M)} \rightarrow [\mathbf{A}, \mathbf{F}] \in \mathbb{R}^{N \times 2} \tag{30}$$

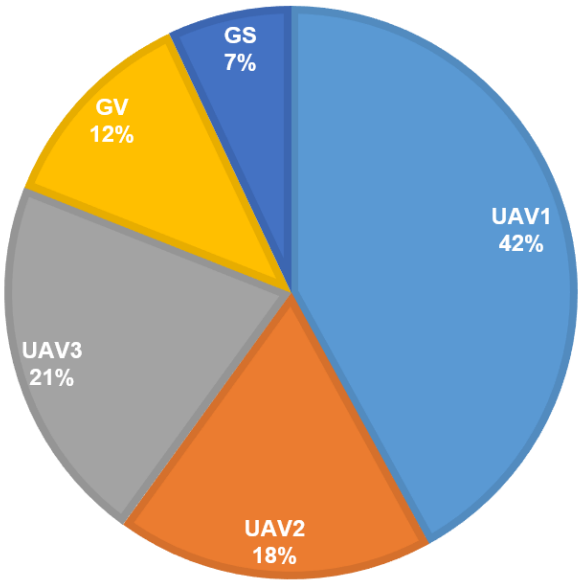

Fig. 3: Roulette wheel selection strategy.

Algorithm 2 U-PSO algorithm

Input: H-MEC positions, UE positions, P, ρ, w_{max}, w_{min}, T_{pso}.

Output: $\boldsymbol{p}_g$.

1: Calculate the dynamic fuzzy membership matrix $\mathbf{U}$ using Eq. (28).
2: Initialize the particle population by using the U-based roulette wheel selection strategy according to $\mathbf{U}$.
3: **while** $t < T_{pso}$ **do**
4: Calculate the fitness value of each particle using Eq. (19).
5: Check the constraints and assign the penalty ρ to the particle with constraint violations.
6: Save the global best particle $\boldsymbol{p}_g$ of the swarm.
7: Save the individual best $\boldsymbol{p}_{bi}$ of each particle.
8: Update the velocity of each particle using Eq. (23) and Eq. (25).
9: Update the position of each particle using Eq. (24).
10: Round the $\boldsymbol{x}_a$ part of each particle.
11: $t = t + 1$.
12: **end while**

where $a_i = 0$ means the i-th UE decides to execute the task itself, $a_i = k$ means the i-th UE decides to offload the task to the k-th H-MEC, while $k \in \{1, 2, \cdots, J + K + M\}$. $f_i \in \mathbb{R}$ means the allocated resource to the i-th UE. If $a_i = 0, f_i = f_i^L$.

Considering that DNN is a universal function approximator, we use DNN to learn the unknown function mapping. The DNN with L layers describes a mapping $f(\mathbf{r}_0; \boldsymbol{\theta}) \in \mathbb{R}^{N_0} \rightarrow \mathbf{r}_L \in \mathbb{R}^{N_L}$ of an input vector $\mathbf{r}_0 \in \mathbb{R}^{N_0}$ to an output vector $\mathbf{r}_L \in \mathbb{R}^{N_L}$ through L iterative processing steps:

$$\mathbf{r}_\ell = f_\ell(\mathbf{r}_{\ell-1}; \boldsymbol{\theta}_\ell), \ell = 1, \cdots, L \tag{31}$$

where $\mathbf{r}_\ell \in \mathbb{R}^{N_\ell}$ is the output of the ℓ-th layer. $\boldsymbol{\theta}_\ell$ is a set of parameters of the ℓ-th layer. The ℓ-th layer is called fully-connected if $f_\ell(\mathbf{r}_{\ell-1}; \boldsymbol{\theta}_\ell)$ has the form

$$f_\ell(\mathbf{r}_{\ell-1}; \boldsymbol{\theta}_\ell) = \sigma(\mathbf{W}_\ell \mathbf{r}_{\ell-1} + \mathbf{b}_\ell) \tag{32}$$

where $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ is the weights of the ℓ-th layer, $\mathbf{b}_\ell \in \mathbb{R}^{N_\ell}$ is the thresholds of the ℓ-th layer. The set of parameters for this layer is $\boldsymbol{\theta}_\ell = \{\mathbf{W}_\ell, \mathbf{b}_\ell\}$. We use $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L\}$ to denote the set of all parameters of the DNN. $\sigma(\cdot)$ is an activation function. Some commonly used activation functions are ReLU, tanh, sigmoid and softmax listed in TABLE II [26].

TABLE II: activation functions of DNN.

Name	$\sigma(v_i)$	Range
ReLU	$max(0, v_i)$	$[0, \infty)$
tanh	$\frac{e^{v_i} - e^{-v_i}}{e^{v_i} + e^{-v_i}}$	$(-1, 1)$
sigmoid	$\frac{1}{1+e^{-v_i}}$	$(0, 1)$
softmax	$\frac{e^{v_i}}{\sum_j e^{-v_i}}$	$(0, 1)$

TABLE III: loss functions of DNN

Name	$\mathcal{L}(\boldsymbol{p}, \boldsymbol{r})$
MSE	$\|\|\boldsymbol{p} - \boldsymbol{r}\|\|_2^2$
Categorical cross-entropy	$-\sum_j p_j \log(r_j)$

In order for the DNN to learn the desired input-output mapping, it is necessary to tune the parameters $\boldsymbol{\theta}$ in a supervised learning fashion. We define the loss function $\mathcal{L}(\boldsymbol{p}^{(n)}, \boldsymbol{r}_L^{(n)})$ that is used to measure the loss between the desired output $\boldsymbol{p}^{(n)}$ and the actual output $\boldsymbol{r}_L^{(n)}$. The goal of the training process is to tune the network parameters $\boldsymbol{\theta}$ in order to minimize the average loss, defined by

$$L(\boldsymbol{\theta}) = \frac{1}{N_t} \sum_{n=1}^{N_t} \mathcal{L}(\boldsymbol{p}^{(n)}, \boldsymbol{r}_L^{(n)}) \tag{33}$$

where N_t is the number of DNN samples. Several relevant loss functions are represented in TABLE III [27]. This minimization problem can be tackled by (stochastic) gradient descent methods, i.e. iteratively updating the parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$ according to the formulas:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \beta \nabla L(\boldsymbol{\theta}(t)) \tag{34}$$

where β is the learning rate, and the gradients are conveniently estimated based on random subsets of the complete training set, called mini-batches and leveraging the back-propagation algorithm [26].

Once the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ to be used are determined as a result of the training process, the DNN is configured and able to compute task admission and resource allocation directly. This means that once the small-scale channel information is changing, the task admission and resource allocation of UEs are updated by a simple feedforward computing of the DNN through performing some simple algebraic calculations, instead of solving Problem $P1$ through high-complexity heuristic algorithms. This yields a significant complexity reduction.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2954503, IEEE Internet of Things Journal

However, there are still two open problems in the design of DNN model for our system. First, the number of UEs accessing the network is time-varying, so the size of $\mathbf{U}$ is ever-changing and we cannot obtain a group of unified samples with the same dimensionality. Second, the Problem $P1$ is a constrained optimization problem, but traditional DNN cannot guarantee the task constraints. With this regard, we propose a novel DNN with a scheduling layer. The main improvements of this new network are listed as follows.

Firstly, to model an efficient DNN for large-scale UEs, we change function mapping $\mathcal{F}$ to $\mathcal{F}1$:

$$\mathcal{F}1\colon \boldsymbol{u}_i \in \mathbb{R}^{(J+K+M)} \rightarrow [a_i, f_i] \in \mathbb{R}^2, \forall i \in \mathcal{N} \tag{35}$$

where $\boldsymbol{u}_i = [u_{i1}, u_{i2}, ..., u_{ic}]$ represents the membership vector of the i-th UE, a_i and f_i represents the task admission value and resource allocation value of the i-th UE, respectively. $a_i = 0$ means the i-th UE decides to execute the task itself, $a_i = k$ means the i-th UE decides to offload the task to the k-th H-MEC, while $k \in \{1, 2, \cdots, J+K+M\}$, $f_i \in \mathbb{R}$ means the allocated resource to the i-th UE. If $a_i = 0, f_i = f_i^L$. This modification has the following benefits. The input dimension of the DNN only depends on the number of H-MECs and is not related to the number of UEs. In general, the number of access points or H-MECs changes much slower than the number of UEs accessing the network. Hence, our DNN can be applied in a long time once it has been trained, which is more practical. Reducing the dimension of input data can also reduce the training complexity, which is suitable for large-scale networks with large number of UEs.

Secondly, in order to guarantee task constraints, we propose a novel DNN structure with a scheduling layer after DNN training. As shown in Fig. 4, this scheduling layer includes a constraint layer and a decision layer, the constraint layer is used to check whether the output a_i and f_i satisfy the constraints or not and each node in this layer represents a constraint check. The output of the j-th node in the constraint layer can be represented as:

$$\mathbf{r}_{L+1,j} = g_j(a_i, f_i) \tag{36}$$

where g_j is the j-th constraint check function. If the output layer of the DNN satisfy the constraint, the function outputs '1' to the next layer, else outputs '0' to the next layer.

The node in the decision layer is labeled as Π, indicating that they play the role of a simple multiplier. If the output layer of the DNN doesn't satisfy all constraints, the final output $\mathbf{r}_{L+2} = 0$, which means UE execute the task locally ($a_i = 0, f_i = f_i^L$), else the final output of the DNN is $\mathbf{r}_{L+2} = \mathbf{r}_L$. The decision layer can be represented as:

$$\mathbf{r}_{L+2} = \mathbf{r}_L \cdot \Pi_{j=1}^{N_c} \mathbf{r}_{L+1,j} \tag{37}$$

where N_c is the number of constraints. The detailed process of the DNN with the scheduling layer is shown in **Algorithm 3**.

The online implementation of the algorithm is as follows. Once the DNN is trained by the supervised learning method. Each UE only needs to input its fuzzy membership values into the DNN, and it will output the resource allocation solution with some simple algebraic calculations, which is much faster than the high-complexity heuristic algorithms. Once one new UE is accessing the network, we need to execute the FCM algorithm to obtain the new locations of the H-MECs and then this UE calculates its fuzzy membership values based on the small-scale channel information and instantaneous interference power. Then, this UE inputs the calculated membership values into the trained DNN, and it will output the resource allocation solution. Hence, our proposed algorithm is very suitable for dynamic scenarios where the number of UEs is varying, which is more practical for implementation.

In addition, the feedback mechanism is applied to H2O framework for online implementation. In practical terms, we use a differential check method to realize the feedback mechanism. When the new data is inputted to the DNN, we check the difference between the new data and the existing samples in the database, which is evaluated by the Euclidean distance. If there is a big difference between the new data and the existing samples, the new data will be fed back to the U-PSO and resolved as a new sample.

Algorithm 3 DNN with a scheduling layer algorithm

Input: $\mathbf{U}$, β, N, T_{CNN}.
Output: $\boldsymbol{\theta}$, $\mathbf{r}_{L+2}$.

Training stage :
Randomly initialize network parameters $\boldsymbol{\theta}$.
while $t < T_{CNN}$ **do**
  Calculate the feedforward of DNN according to Eqs. (31)-(32) for all layers.
  Calculate the loss function according to Eq. (33).
  Update of DNN according to Eq. (34).
  $t = t + 1$.
end while
Serialize $\mathbf{U}$ into a set of $\boldsymbol{u}_i$ according to the UE number N.

Decision stage :
while $i < N$ **do**
  Calculate the output a_i and f_i of the DNN based on the trained $\boldsymbol{\theta}$ according to the input $\boldsymbol{u}_i$.
  Check the output of DNN by the constraint layer according to Eq. (36).
  Calculate the decision output $\mathbf{r}_{L+2}$ by the decision layer according to Eq. (37).
  $i = i + 1$.
end while

IV. Simulation results

In this section, we use simulations to evaluate the performance of the proposed H2O algorithm. In all simulations, we use three UAVs, one GV and one GS. Other parameters used in the simulations are summarized in TABLE IV, unless otherwise specified. The parameters of the LS-FCM method are chosen as follows: $\tau = 2$, $\varepsilon = 0.0001$, $T_{FCM} = 100$; The parameters of the U-PSO method are chosen as follows: $P = 10, \rho = 100, w_{max} = 0.9, w_{min} = 0.4, T_{PSO} = 100$; The parameters of the DNN method are chosen as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2954503, IEEE Internet of Things Journal

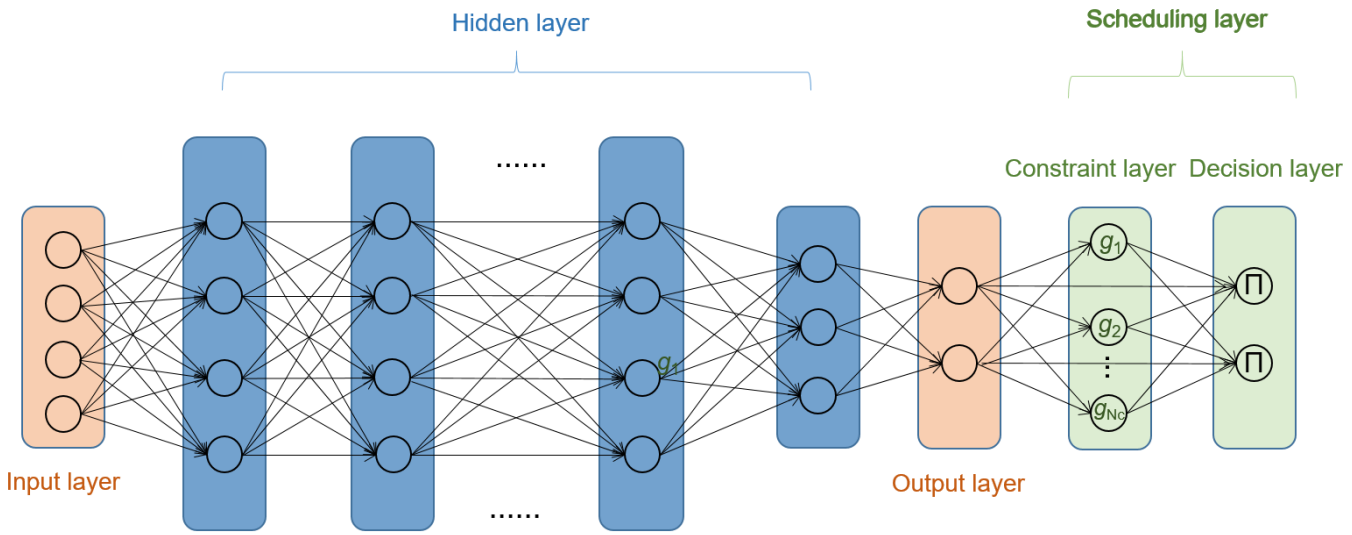

Fig. 4: DNN with the scheduling layer.

TABLE IV: Simulation parameters

Parameters	Assumptions
Macrocell Radius	100m
GS Coordinate	[50,50]
Bandwidth B	1MHz
Transmitting Power P_{ij}^{T}	1W
Input Data Size D_i	100kB
Required Number of CPU Cycles f_i^L	10^9 cycles/s
Local Computational Capability F_{max}^{L}	10^9 cycles/s
Latency Request T^{req}	2s
UAV Altitude H^{UAV}	20m
Remote Computational Capability (UAV)	10^{10} cycles/s
Remote Computational Capability (GV)	10^{11} cycles/s
Remote Computational Capability (GS)	10^{12} cycles/s

$\beta = 0.4, T_{DNN} = 500$, the activation function is set to ReLU and function is set to MSE.

Apart from the proposed task admission algorithm, we also simulate the following algorithms for comparison purposes.

- Random offloading (Random): the task admission is decided randomly for each UE. If the computational resources of the allocated H-MEC are insufficient, UE execute the task locally.
- Greedy offloading (Greedy): all UEs offload the task to the nearest H-MEC, if the computational resources of the allocated H-MECs are insufficient, the UEs who need more computational resources of the H-MECs execute the task locally.
- Local execution (Local): There is no offloading. All tasks are executed locally.
- PSO offloading (PSO): the task admission is optimized by the U-PSO method.

All sample vectors in the dataset are split randomly into a training set and a testing set. 80 percent of the sample vectors are assigned into the training set, while 20 percent of the sample vectors are assigned into the testing set. Then cross validation method [28] is used to evaluate the performance of DNN. In Fig. 5, we compare the performance of DNN with different number of hidden nodes in hidden layer, as the number of hidden layers varies from 1 to 8. With the increase of hidden layer number, the testing loss of DNN with 20 hidden nodes and 30 hidden nodes first decrease and then stabilize when the hidden layer number is above 6. This is because increasing hidden nodes and hidden layers can enhance the learning ability of DNN and allow the DNN to learn more information. The DNN with 30 hidden nodes of each hidden layer achieves the minimum testing loss when the hidden layer number is above 6. Notice that the testing loss of DNN with 10 hidden nodes first decrease and then increase when the hidden layer number is above 4. This is due to the concept of overfitting and the DNN cannot improve its learning ability when the hidden nodes is too less. For this reason, in the follow simulations, we set the hidden layer number equal 6, the hidden nodes number equal 30.

In Fig. 6, we plot the training loss and testing loss of DNN. It is clearly seen that the testing loss declines sharply at the beginning of procedure, then decreases slowly and stabilize at around 0.07 when t is above 300. Meanwhile, the training loss gradually decreases and stabilizes at around 0.06, whose value is less than the testing loss curve. In Fig. 7, we further study the error distribution of all samples for the DNN. We see that 70% of the training errors are less than 0.025 and the maximum error is less than 0.47. The same situation applies to the error distribution of the testing samples. The simulation results demonstrate the effectiveness of the proposed DNN which can quickly converge to an optimal decision model for our offloading problem.

In Fig. 8, we compare the performance of DNN trained by different samples under varying time slots. Before the evaluation, DNN has been trained with 10000 independent

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2954503, IEEE Internet of Things Journal

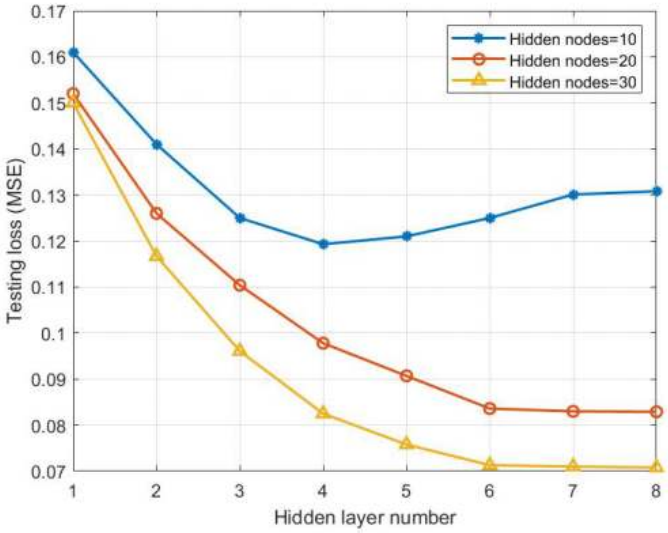

Fig. 5: The comparison of testing loss of DNN with different hidden layer number and different hidden nodes number.

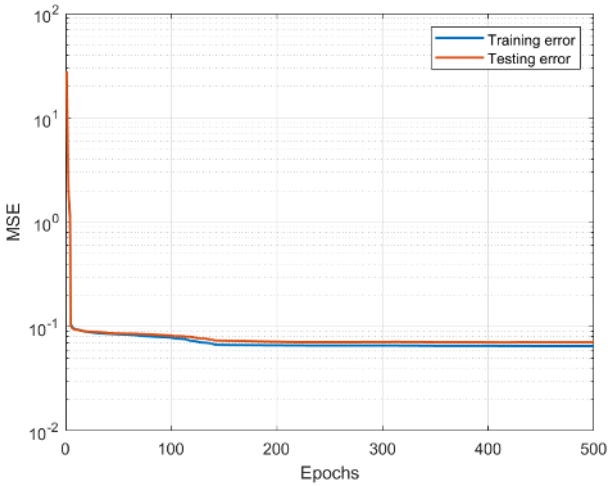

Fig. 6: The testing loss and training loss of DNN.

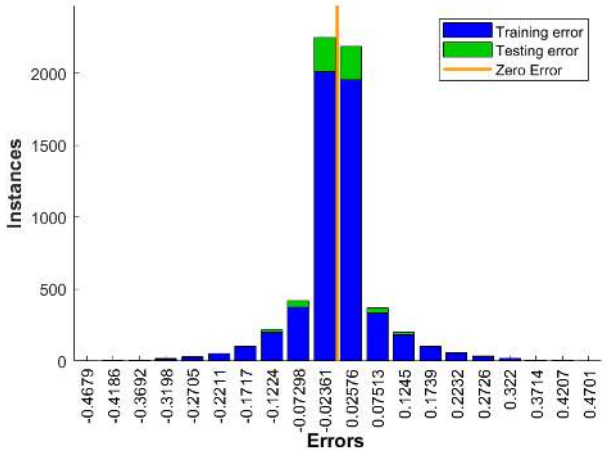

Fig. 7: The training and testing error distribution of all samples for the DNN.

samples generated from different algorithms (the proposed PSO, Greedy and Random), and the error curve has converged. We see that DNN achieves optimal performance with the samples generated from the proposed PSO (PSO+DNN), and significantly outperforms the DNN with the greedy offloading samples (Greedy+DNN) and the random offloading samples (Random+DNN). This is because the performance of DNN depends on the quality of samples. The proposed PSO can solve the offloading problem by a heuristic global search and achieve a high quality global solution, which guarantee the good performance of DNN from the learning stage.

We then evaluate the performance of 30 independent random scenarios for time slots prediction. We see that DNN with the samples generated from the proposed PSO can achieve the best performance under all time slots. This is because the trained DNN has good generalization performance, and can make the offloading decision continuously in fast changing environments.

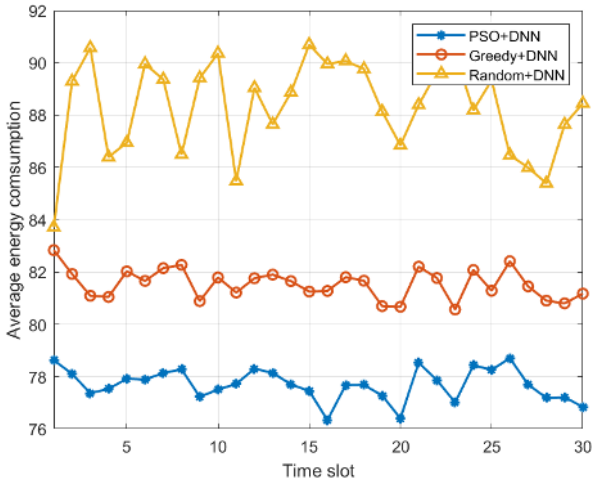

Fig. 8: The comparison of average energy consumption of DNN trained by different samples under varying time slots.

In Fig. 9, we compare the average energy consumption between the proposed method, PSO, Greedy, Random and Local, as the number of devices varies from 10 to 100. As shown in the Fig. 9, the average energy consumptions of all methods increase gradually while the number of devices increases. PSO method achieves the lowest average energy consumption. The proposed method provides almost the same energy consumption compared as PSO. This is because the proposed method is a DNN model trained by a set of high quality global solutions generated by PSO and construct a nonlinear mapping from UE and MEC information to offloading decision. Greedy method saves more energy than Random method. The energy consumption of Local is highest, since Local does not admit any devices.

Fig. 10 shows the number of admitted devices (the devices who decide to offload the tasks) achieved by the proposed method, Greedy, Random and Local, where the number of devices range from 10 to 100. We can see that the proposed method admit the most devices for offloading. Greedy can

admit more devices for offloading than Random. In contrast, Local does not admit any device. Fig. 11 shows the number of admitted devices achieved by the proposed method, Greedy, Random and Local, where T^{req} ranges from 1s to 3s. We can see the similar results again, there may be some interesting conclusions that, for achieving the goal of minimum energy, the number of admitted devices increases as the number of devices varies from 10 to 100 or T^{req} varies from 1s to 3s.

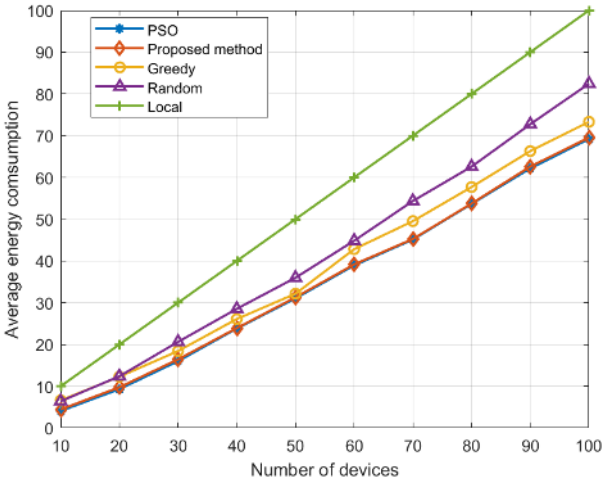

Fig. 9: The comparison of average energy consumption as the number of devices varies from 10 to 100.

Finally, we evaluate the computation time of the proposed algorithm. The computational complexity of all algorithms greatly depends on the number of uses. Fig. 12 compares runtime between the proposed method and PSO. We can see the proposed method is superior in terms of time efficiency. In particular, it generates an offloading action in less than 0.03 second when N equals to 100, while PSO takes 80 times longer CPU time. Overall, trained DNN achieves similar performance as PSO algorithm but requires substantially less CPU time. This makes real-time offloading and resource allocation viable for H-MEC networks in fast changing environment.

V. Conclusion

In this paper, we have proposed a hybrid deep learning based online offloading algorithm, H2O, to minimize the sum energy consumption of UEs in a H-MEC network with binary computation offloading. The framework includes three artificial intelligence algorithms: an LS-FCM method is used to locate the GVs and UAVs, a U-PSO is used to solve the MINLP problem and provide high quality samples to DNN, a DNN with a scheduling layer is applied to make the task admission and resource allocation decision in real time. Compared to conventional optimization methods, the proposed H2O algorithm completely removes the need of solving MINLP problems in the decision period of DNN. Simulation results show that H2O achieves similar near-optimal performance as heuristic methods but reduces the CPU time by more than several orders of magnitude, making real-time system optimization feasible for the H-MEC networks in fast-changing environment.

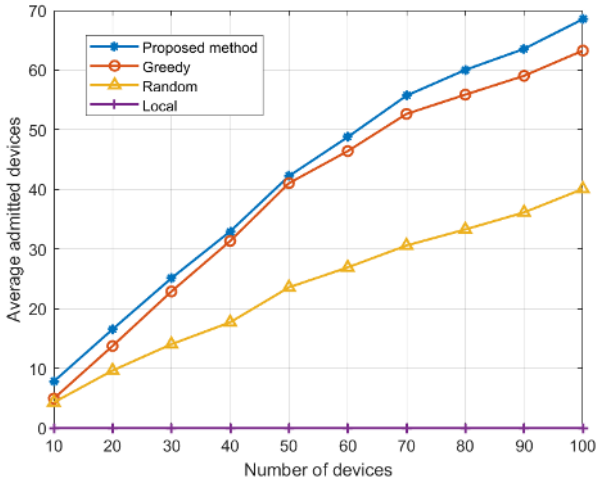

Fig. 10: The comparison of average admitted devices as the number of devices varies from 10 to 100.

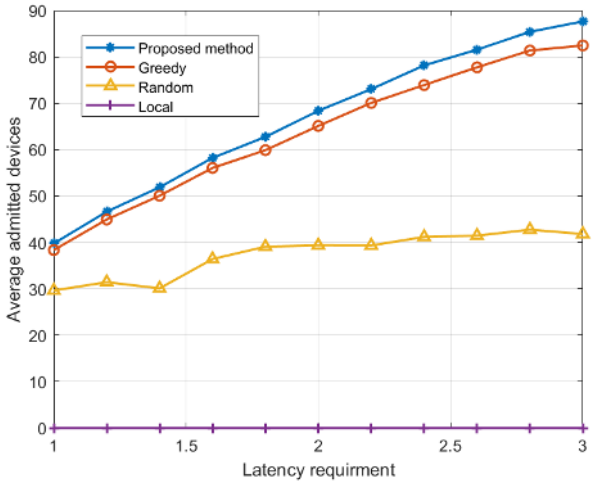

Fig. 11: The comparison of average admitted devices as T^{req} varies from 1s to 3s.

References

[1] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, July 2012, pp. 59–66.

[2] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *IEEE Network*, vol. 27, no. 5, pp. 34–40, Sep. 2013.

[3] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE J. Select. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov 2017.

[4] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2429–2445, Nov 2018.

[5] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[6] Y. Du, K. Wang, K. Yang, and G. Zhang, "Energy-efficient resource allocation in uav based mec system for iot devices," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec 2018, pp. 1–6.

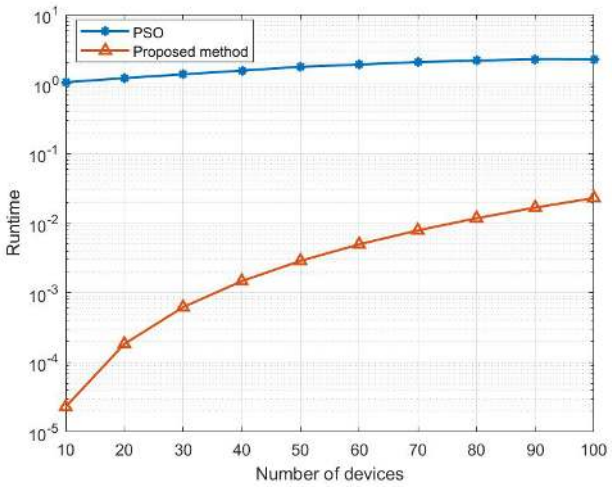

Fig. 12: The comparison of runtime between PSO and the proposed method as the number of devices varies from 10 to 100.

[7] Narendra and Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Trans. Comput.*, vol. C-26, no. 9, pp. 917–922, Sep. 1977.

[8] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 1, no. 2.

[9] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, June 2018.

[10] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan 2019.

[11] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE Int. Conf. on Comput. Commun.(INFOCOM)*, April 2016, pp. 1–9.

[12] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug 2017.

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

[14] H. Jiang, D. Peng, K. Yang, Y. Zeng, and Q. Chen, "Predicted mobile data offloading for mobile edge computing systems," in *Proc. Int. Conf. on Smart Comput. and Commun.* Springer, 2018, pp. 153–162.

[15] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Select. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[16] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Netw. Appl.*, Nov 2018. [Online]. Available: https://doi.org/10.1007/s11036-018-1177-x

[17] C. H. Liu, Z. Chen, and Y. Zhan, "Energy-efficient distributed mobile crowd sensing: A deep learning approach," *IEEE J. Select. Areas Commun.*, vol. 37, no. 6, pp. 1262–1276, June 2019.

[18] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, Jan 2018.

[19] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 760–770, July 2018.

[20] L. Yang, J. Cao, S. Tang, T. Li, and A. T. S. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, June 2012, pp. 794–802.

[21] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing." *HotCloud*, vol. 10, no. 4, pp. 4–19, 2010.

[22] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, April 2015.

[23] H. He, S. Zhang, Y. Zeng, and R. Zhang, "Joint altitude and beamwidth optimization for uav-enabled multiuser communications," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 344–347, Feb 2018.

[24] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Machine and Human Science*, Oct 1995, pp. 39–43.

[25] F. Jiang, Q. Dai, and L. Dong, "An icpso-rbfnn nonlinear inversion for electrical resistivity imaging," *J. Cent. South Univ.*, vol. 23, no. 8, pp. 2129–2138, Aug 2016.

[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Dec 2015.

[27] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 3, no. 4, pp. 563–575, Dec 2017.

[28] F. Jiang, L. Dong, and Q. Dai, "Electrical resistivity imaging inversion: An isfla trained kernel principal component wavelet neural network approach," *Neural Networks*, vol. 104, pp. 114–123, Aug 2018.

Feibo Jiang received his B.S. and M.S. degrees in School of Physics and Electronics from Hunan Normal University, China, in 2004 and 2007, respectively. He received his Ph.D. degree in School of Geosciences and Info-physics from the Central South University, China, in 2014. He is currently an associate professor at the Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, China. His research interests include artificial intelligence, fuzzy computation, Internet of Things, and mobile edge computing.

Kezhi Wang received his B.E. and M.E. degrees in School of Automation from Chongqing University, China, in 2008 and 2011, respectively. He received his Ph.D. degree in Engineering from the University of Warwick, U.K. in 2015. He was a Senior Research Officer in University of Essex, U.K. Currently he is a Senior Lecturer in Department of Computer and Information Sciences at Northumbria University, U.K. His research interests include wireless communication, mobile edge computing, UAV communication and machine learning.

Li Dong received the B.S. and M.S. degrees in School of Physics and Electronics from Hunan Normal University, China, in 2004 and 2007, respectively. She received her Ph.D. degree in School of Geosciences and Info-physics from the Central South University, China, in 2018. Her research interests include machine learning, Internet of Things, and mobile edge computing.

Cunhua Pan received the B.S. and Ph.D. degrees from the School of Information Science and Engineering, Southeast University, Nanjing, China, in 2010 and 2015, respectively. From 2015 to 2016, he was a Research Associate at the University of Kent, U.K. He held a post-doctoral position at Queen Mary University of London, U.K., from 2016 and 2019, where he is currently a Lecturer. His research interests mainly include ultra-dense C-RAN, machine learning, UAV, Internet of Things, and mobile edge computing. He serves as a TPC member for numerous conferences, such as ICC and GLOBECOM, and the Student Travel Grant Chair for ICC 2019. He also serves as an Editor of IEEE ACCESS.

Wei Xu (S'07-M'09-SM'15) received his B.Sc. degree in electrical engineering and his M.S. and Ph.D. degrees in communication and information engineering from Southeast University, Nanjing, China in 2003, 2006, and 2009, respectively. Between 2009 and 2010, he was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Victoria, Canada. He is currently a Professor at the National Mobile Communications Research Laboratory, Southeast University. He is also an Adjunct Professor of the University of Victoria in Canada, and a Distinguished Visiting Fellow of the Royal Academy of Engineering, U.K. He has co-authored over 100 refereed journal papers in addition to 36 domestic patents and four US patents granted. His research interests include cooperative communications, information theory, signal processing and machine learning for wireless communications. He was an Editor of the IEEE COMMUNICATIONS LETTERS from 2012 to 2017. He is currently an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS and the IEEE ACCESS. He received the Best Paper Awards from IEEE MAPE 2013, IEEE/CIC ICCC 2014, IEEE Globecom 2014, IEEE ICUWB 2016, WCSP 2017, and ISWCS 2018. He was the co-recipient of the First Prize of the Science and Technology Award in Jiangsu Province, China, in 2014. He received the Youth Science and Technology Award of China Institute of Communications in 2018.

Kun Yang received his PhD from the Department of Electronic & Electrical Engineering of University College London (UCL), UK. He is currently a Chair Professor in the School of Computer Science & Electronic Engineering, University of Essex, leading the Network Convergence Laboratory (NCL), UK. Before joining in the University of Essex at 2003, he worked at UCL on several European Union (EU) research projects for several years. His main research interests include wireless networks and communications, IoT networking, data and energy integrated networks, mobile edge computing. He manages research projects funded by various sources such as UK EPSRC, EU FP7/H2020 and industries. He has published 150+ journal papers and filed 10 patents. He serves on the editorial boards of both IEEE and non-IEEE journals. He is a Senior Member of IEEE (since 2008) and a Fellow of IET (since 2009).