Resource Awareness in Unmanned Aerial Vehicle-Assisted Mobile-Edge Computing Systems

Xianfu Chen*, Tao Chen*, Zhifeng Zhao†, Honggang Zhang‡, Mehdi Bennis§, and Yusheng Ji¶
*VTT Technical Research Centre of Finland Ltd, Finland
†Research Center for Intelligent Networks, Zhejiang Lab, Hangzhou, China
‡College of Information Science and Electronic Engineering, Zhejiang University, China
§Centre for Wireless Communications, University of Oulu, Finland
¶Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, Japan

***Abstract*—This paper investigates an unmanned aerial vehicle (UAV)-assisted mobile-edge computing (MEC) system, in which the UAV provides complementary computation resource to the terrestrial MEC system. The UAV processes the received computation tasks from the mobile users (MUs) by creating the corresponding virtual machines. Due to finite shared I/O resource of the UAV in the MEC system, each MU competes to schedule local as well as remote task computations across the decision epochs, aiming to maximize the expected long-term computation performance. The non-cooperative interactions among the MUs are modeled as a stochastic game, in which the decision makings of a MU depend on the global state statistics and the task scheduling policies of all MUs are coupled. To approximate the Nash equilibrium solutions, we propose a proactive scheme based on the long short-term memory and deep reinforcement learning (DRL) techniques. A digital twin of the MEC system is established to train the proactive DRL scheme offline. Using the proposed scheme, each MU makes task scheduling decisions only with its own information. Numerical experiments show a significant performance gain from the scheme in terms of average utility per MU across the decision epochs.**

***Index Terms*—Mobile-edge computing, unmanned aerial vehicle, resource awareness, deep reinforcement learning, long short-term memory, digital twin.**

I. Introduction

Mobile-edge computing (MEC), which provides computing capabilities within the radio access networks (RANs) in close proximity to the mobile users (MUs), is a promising paradigm to address the tension between computation-intensive applications and resource-constrained mobile devices [1]. By offloading computation tasks to the resource-rich MEC cloud, not only the computation qualities of service and experience can be greatly improved, but also the capability of a mobile device can be augmented for running a variety of resource-demanding applications. Recently, there are a number of related works on designing computation offloading schemes. For example, in [2], Wang et al. proposed a Lagrangian duality method to minimize the total energy consumption in a computation latency constrained wireless powered multiuser MEC system. In [3], Liu et al. studied the power-delay tradeoff for a MEC system using the Lyapunov optimization technique. In our priori work [4], the infinite time-horizon Markov decision process (MDP) framework was used to model the problem of computation offloading for a MU in an ultra-dense RAN and to solve the optimal policies, we proposed the deep reinforcement learning (DRL) based schemes.

Offloading the input data of a task from the mobile device of a MU to the MEC cloud requires wireless transmissions, which account for the dynamics from the surrounding environment. Particularly, the time-varying channel qualities due to the MU mobility in turn limits the computation performance [5]. Because of among others, the low deployment cost, the flexibility and the line-of-sight (LOS) connections, unmanned aerial vehicles (UAVs) are expected to play a significant role in advancing the future wireless networks [6]. Leveraging the UAV technology in a MEC system has been shown to be substantial. In [7], Hu et al. put forward an alternating algorithm to minimize the weighted sum energy consumption for a UAV-assisted MEC system. In [8], Zhou et al. investigated a UAV-enabled wireless-powered MEC system and derived alternating algorithms to solve the computation rate maximization problems under both the partial and the binary computation offloading modes. However, most of the existing literature is basically based on a finite time-horizon.

In this paper, we concentrate on a three-dimensional UAV-assisted MEC system, in which a UAV is implemented as a complementary computing server flying in the air. That is, in addition to local computation execution, each MU in the system can also offload a computation task to the UAV or to the MEC cloud via one of the base stations (BSs) in the RAN. The UAV can co-execute the computation tasks of the MUs by creating isolated virtual machines (VMs) [9]. Sharing the same physical UAV platform causes I/O interference, leading to computation rate reduction for each VM. Under this context, the MUs compete to schedule local and remote task computations with the awareness of environmental dynamics. The aim of each MU is to maximize the expected long-term computation performance. The non-cooperative interactions among the MUs are modeled as a stochastic game. Solving a Nash equilibrium (NE) of the stochastic game needs complete information exchange among the MUs, which is practically overwhelming. Motivated by recent advances in recurrent and deep neural networks, we propose a proactive DRL scheme, enabling each MU to behave at an approximated NE only with local information [10], [11]. Furthermore, we establish a digital twin of the MEC system to get over the hurdle of training the neural networks [12]. To the best of our knowledge, there does not exist a comprehensive study on stochastic resource awareness among the non-cooperative MUs in a UAV-assisted MEC system.

978-1-7281-5207-3/20/$31.00 ©2020 IEEE

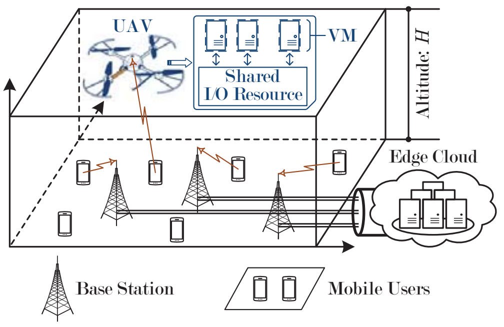

Fig. 1. Illustration of an unmanned aerial vehicle (UAV)-assisted mobile-edge computing system (VM: virtual machine.).

II. System Descriptions and Assumptions

As illustrated in Fig. 1, we focus on a three-dimensional scenario, in which a terrestrial MEC system is assisted by a UAV. The UAV hovers in the air at a fixed altitude of H (in meters) [1]. The terrestrial MEC system consists of a set $\mathcal{B} = \{1, \cdots, B\}$ of BSs, which are connected via wired links to the computing cloud at the edge. To ease analysis, we use a common finite set $\mathcal{L}$ of locations (i.e., small two-dimensional non-overlapping areas)[2] to denote both the terrestrial service region covered by the BSs and the region of the UAV mapped vertically from the air to the ground. In the system, a set $\mathcal{K}$ of MUs coexist and generate sporadic computation tasks over the infinite time-horizon, which is discretized into decision epochs. Each epoch is assumed to be of equal duration δ (in seconds) and indexed by an integer $j \in \mathbb{N}_+$.

A. Mobility Model

We apply the smooth-turn mobility model with a reflecting boundary to simulate the UAV trajectory [14]. In this model, the UAV maintains a constant forward speed but randomly changes the centripetal acceleration. Let $L^j_{(\text{UAV})} \in \mathcal{L}$ be the mapped terrestrial location of the UAV during a decision epoch j. With regards to the MUs, their movements are modelled using a boundary Gauss-Markov mobility model [15]. Specifically, the location $L^j_{(\text{MU}),k} \in \mathcal{L}$ of each MU $k \in \mathcal{K}$ during each decision epoch j is determined by both the location $L^{j-1}_{(\text{MU}),k}$ at epoch $j-1$ and the velocity during epoch j, while the velocity of a MU during a decision epoch depends on the velocity during the previous epoch only.

B. Task Model

The computation task arrivals at the MUs are assumed to be independent and identically distributed sequences of Bernoulli random variables with a common parameter $\lambda \in [0, 1]$. More specifically, we choose $A^j_k \in \{0, 1\}$ to be the task arrival indicator for a MU $k \in \mathcal{K}$, that is, $A^j_k = 1$ if a computation task is generated at MU k in the end of epoch j and otherwise, $A^j_k = 0$. Then, $\mathbb{P}(A^j_k = 1) = 1 - \mathbb{P}(A^j_k = 0) = \lambda$, $\forall k \in \mathcal{K}$, where $\mathbb{P}(\cdot)$ denotes the probability of the occurrence of an event. We let μ (in bits) and ϑ represent, respectively, the input data size and the number of CPU cycles required to accomplish one input bit of a computation task. The arrived but not processed tasks will be queued at the buffer of a MU. A computation task can be either computed locally at the device of the MU or executed remotely (at the UAV or the MEC cloud). We let $X^j_k \in \{0, 1\}$ and $F^j_k \in \mathcal{B} \cup \{0, B+1\}$ denote the local and remote computation task scheduling decisions of MU k at each decision epoch j. That is, $X^j_k = 1$ if MU k sends a computation task to the local CPU and otherwise, $X^j_k = 0$, while if MU k offloads the computation task to the UAV, $F^j_k = B + 1$, or to the MEC cloud via one of the BSs, $F^j_k = b$ ($b \in \mathcal{B}$) and otherwise, $F^j_k = 0$. Hence the task queue dynamics of MU k can be expressed as

$$Q^{j+1}_k = \max\left\{Q^j_k - X^j_k - \mathbb{1}_{\{F^j_k > 0\}}, 0\right\} + A^j_k, \qquad (1)$$

where Q^j_k is the number of computation tasks in the task buffer of MU k at the beginning of decision epoch j and $\mathbb{1}_{\{\cdot\}}$ is an indicator function that equals 1 if the condition is satisfied and 0, otherwise. In this work, we assume a large enough buffer capacity for a MU to avoid the buffer overflows.

C. Computation Model

The UAV complements the terrestrial MEC system with the computation resource from the air. By strategically offloading the computation tasks to the UAV or the MEC cloud via one of the BSs for remote execution, the MUs can expect a significantly improved computation experience.

1) Local Computation: When a computation task is scheduled for processing locally at the mobile device of a MU $k \in \mathcal{K}$ during a decision epoch j, i.e., $X^j_k = 1$, the number of needed epochs can be calculated as $\Delta = \lceil(\mu \cdot \vartheta)/(\rho \cdot \delta)\rceil$, where $\lceil\cdot\rceil$ means the ceiling function and we assume that the local CPU of a MU operates at frequency ρ (in Hz). We describe the local processing state $S^j_{(\text{MU}),k} \in \{0, 1, \cdots, \Delta\}$ at a decision epoch j using the number $S^j_{(\text{MU}),k}$ of remaining epochs to finish the computation task. For local computation during an epoch j, the processing delay experienced by MU k is given by

$$D^j_{(\text{MU}),k} = \begin{cases} 0, & \text{if } S^j_{(\text{MU}),k} = 0; \\ \dfrac{\mu \cdot \vartheta - (\Delta - 1) \cdot \delta \cdot \rho}{\rho}, & \text{if } S^j_{(\text{MU}),k} = 1; \\ \delta, & \text{if } S^j_{(\text{MU}),k} > 1, \end{cases} \qquad (2)$$

and the resulted energy consumed by the mobile device of MU k then is

$$E^j_{(\text{MU}),k} = \begin{cases} 0, & \text{if } S^j_{(\text{MU}),k} = 0; \\ \tau \cdot (\mu \cdot \vartheta - (\Delta - 1) \cdot \delta \cdot \rho) \cdot (\rho)^2, & \text{if } S^j_{(\text{MU}),k} = 1; \\ \tau \cdot \delta \cdot (\rho)^3, & \text{if } S^j_{(\text{MU}),k} > 1, \end{cases} \qquad (3)$$

where τ is the effective switched capacitance that depends on the chip architecture of a mobile device [16].

[1] This work assumes that the power of the UAV is supplied by laser charging [13]. Hence the UAV is able to operate over the long run.

[2] Each location or small area can be characterized by uniform wireless communication conditions [5].

2) Remote Execution: For remote computation execution, a MU has to be first associated with a BS or the UAV until the task is accomplished. Let $I_k^j \in \mathcal{B} \cup \{B+1\}$ be the association state of each MU $k \in \mathcal{K}$ during a decision epoch j, namely, $I_k^j = b \in \mathcal{B}$ if MU k is associated with a BS b and if MU k is associated with the UAV, $I_k^j = B+1$. Then

$$I_k^j = i \cdot \mathbb{1}_{\{\{F_k^j = i\} \vee \{\{F_k^j = 0\} \wedge \{I_k^{j-1} = i\}\}\}}, \tag{4}$$

where $i \in \mathcal{B} \cup \{B+1\}$, while $\vee$ and $\wedge$ mean, respectively, logic OR and logic AND. When $I_k^j \neq I_k^{j-1}$, which may happen only when $F_k^j > 0$[3], a handover among the BSs and the UAV is hence triggered [4]. We assume that the energy consumption during the occurrence of one handover is negligible at MU k but the incurred delay is ζ (in seconds). During a decision epoch j, MU k experiences the average channel power gains $G_{b,k}^j = g_{(\mathrm{BS})}(L_{(\mathrm{MU}),k}^j, L_{(\mathrm{BS}),b})$ for the link between MU k and BS b and $G_{(\mathrm{UAV}),k}^j = g_{(\mathrm{UAV})}(L_{(\mathrm{MU}),k}^j, L_{(\mathrm{UAV})}^j, H)$ for the link between MU k and the UAV, which are determined by the physical distances.

At the beginning of a decision epoch j, if a MU $k \in \mathcal{K}$ lets the MEC cloud execute a computation task, all input data needs to be offloaded via a BS $F_k^j = b \in \mathcal{B}$, for which the achievable data rate can be written as $R_{b,k}^j = W \cdot \log_2(1 + (G_{b,k}^j \cdot P_k)/(W \cdot \sigma^2))$, where W is the frequency bandwidth exclusively allocated to a MU, P_k is the transmit power and σ^2 is the noise power spectral density. We use $T_{(\mathrm{BS}),k}^j \in [0, \mu]$ to denote the local transmission state of MU k at the beginning of a decision epoch j, which indicates the remaining amount of input data to be transmitted for the task. Hence the transmission delay[4] and the energy consumption during epoch j are calculated as $Y_{(\mathrm{BS}),k}^j = \min\{T_{(\mathrm{BS}),k}^j / R_{b,k}^j + \zeta \cdot \mathbb{1}_{\{I_k^j \neq I_k^{j-1}\}}, \delta\}$ and $E_{(\mathrm{BS}),k}^j = P_k \cdot (Y_{b,k}^j - \zeta \cdot \mathbb{1}_{\{I_k^j \neq I_k^{j-1}\}})$. In this paper, we assume that the BSs are connected using the wired links to the MEC cloud, which is of rich computation resource. We ignore the round-trip delay between the BSs and the MEC cloud as well as the time consumed for processing a computation task at the MEC cloud. Further, the time consumed by the selected BS (or the UAV in the following) to send back the computation result is negligible due to the fact that the size is much smaller than the input data of a computation task [17].

Similarly, if a MU $k \in \mathcal{K}$ offloads a computation task to the UAV for processing at a decision epoch j, namely, $F_k^j = B+1$, the time[5] and the energy consumed during each decision epoch j turn to be $Y_{(\mathrm{UAV}),k}^j = \delta \cdot \mathbb{1}_{\{T_{(\mathrm{UAV}),k}^j > 0\}}$ and $E_{(\mathrm{UAV}),k}^j = P_k \cdot (\min\{T_{(\mathrm{UAV}),k}^j / R_{(\mathrm{UAV}),k}^j + \zeta \cdot \mathbb{1}_{\{I_k^j \neq I_k^{j-1}\}}, \delta\} - \zeta \cdot \mathbb{1}_{\{I_k^j \neq I_k^{j-1}\}})$, respectively, where $R_{(\mathrm{UAV}),k}^j = W \cdot \log_2(1 + (G_{(\mathrm{UAV}),k}^j \cdot P_k)/(W \cdot \sigma^2))$ is the achievable data rate, while $T_{(\mathrm{UAV}),k}^j \in [0, \mu]$ denotes the transmission state at a decision epoch j. Let $\mathcal{K}^j \subseteq \mathcal{K}$ represent the subset of MUs, whose computation tasks are being simultaneously processed by the corresponding VMs at the UAV during a decision epoch j. Denote by C_0 the computation service rate of a VM at the UAV given that the task is run in isolation, the degraded computation rate of each MU $k \in \mathcal{K}^j$ is modeled as $C^j = C_0 \cdot (1 + \varphi)^{1 - |\mathcal{K}^j|}$, where $|\cdot|$ means the cardinality of a set and $\varphi \in \mathbb{R}_+$ is a factor specifying the percentage of reduction in the computation rate of a VM when multiplexed with another VM. Accordingly, we obtain the remote processing delay of MU k during decision epoch j as $D_{(\mathrm{UAV}),k}^j = \min\{S_{(\mathrm{UAV}),k}^j / C^j, \delta\}$ with the remote processing state $S_{(\mathrm{UAV}),k}^j \in [0, \mu]$ showing the amount of input data to be processed at the beginning of an epoch j.

III. Problem Formulation and Game-Theoretic Solution

During each decision epoch j, the local state of a MU $k \in \mathcal{K}$ can be described by $\boldsymbol{\xi}_k^j = (L_{(\mathrm{MU}),k}^j, L_{(\mathrm{UAV})}^j, Q_k^j, I_k^j, S_{(\mathrm{MU}),k}^j, S_{(\mathrm{UAV}),k}^j, T_{(\mathrm{BS}),k}^j, T_{(\mathrm{UAV}),k}^j) \in \mathcal{Z}$, where $\mathcal{Z}$ is a common finite state space for all MUs. We use $\boldsymbol{\xi}^j = (\boldsymbol{\xi}_k^j, \boldsymbol{\xi}_{-k}^j) \in \mathcal{Z}^{|\mathcal{K}|}$ to represent the global system state with $-k$ denoting all the other MUs in $\mathcal{K}$ without the presence of a MU k. Let π_k be the stationary task scheduling policy employed by MU k. When deploying π_k, MU k observes $\boldsymbol{\xi}^j$ at the beginning of a decision epoch j and accordingly, makes local as well as remote task scheduling decisions, that is, $\pi_k(\boldsymbol{\xi}^j) = (X_k^j, F_k^j)$. We define an immediate utility function[6]

$$u_k\left(\boldsymbol{\xi}^j, \left(X_k^j, F_k^j\right)\right) = \exp\left(-D_k^j\right) + \eta \cdot \exp\left(-E_k^j\right), \tag{5}$$

to measure the satisfaction of experienced delay and consumed energy for each MU k during each epoch j, where $\eta \in \mathbb{R}_+$ is the weighting constant, $D_k^j = D_{(\mathrm{MU}),k}^j + D_{(\mathrm{UAV}),k}^j + Y_{(\mathrm{BS}),k}^j + Y_{(\mathrm{UAV}),k}^j + \delta \cdot \max\{Q_k^j - X_k^j - \mathbb{1}_{\{F_k^j > 0\}}, 0\}$ is composed of not only the processing and transmission delay but also the task queueing delay, while $E_k^j = E_{(\mathrm{MU}),k}^j + E_{(\mathrm{BS}),k}^j + E_{(\mathrm{UAV}),k}^j$ constitutes the total local energy consumption.

Along with the discussions, it can be easily verified that the randomness lying in a sequence of the global system states over the time horizon $\{\boldsymbol{\xi}^j : j \in \mathbb{N}_+\}$ is Markovian. Given a stationary task scheduling policy π_k by each MU $k \in \mathcal{K}$ and an initial global state $\boldsymbol{\xi}^1 = \boldsymbol{\xi} \in \mathcal{Z}^{|\mathcal{K}|}$, we express the expected long-term discounted utility function $V_k(\boldsymbol{\xi}, (\pi_k, \boldsymbol{\pi}_{-k}))$ of MU k as

$$V_k(\boldsymbol{\xi}, (\pi_k, \boldsymbol{\pi}_{-k})) = \tag{6}$$

$$(1-\gamma) \cdot \mathsf{E}_{(\pi_k, \boldsymbol{\pi}_{-k})}\left[\sum_{j=1}^{\infty} (\gamma)^{j-1} \cdot u_k\left(\boldsymbol{\xi}^j, \left(X_k^j, F_k^j\right)\right) | \boldsymbol{\xi}^1 = \boldsymbol{\xi}\right],$$

where $\gamma \in [0, 1)$ is the discount factor and the expectation $\mathsf{E}_{(\pi_k, \boldsymbol{\pi}_{-k})}[\cdot]$ is taken over different decision makings under different global system states following a joint task scheduling policy $(\pi_k, \boldsymbol{\pi}_{-k})$ across the decision epochs. When γ approaches 1, (6) approximates the expected long-term undiscounted utility as well [18]. $V_k(\boldsymbol{\xi}, (\pi_k, \boldsymbol{\pi}_{-k}))$ is also termed

[3] If a MU $k \in \mathcal{K}$ does not offload a task at the beginning of a decision epoch j, the association state remains unchanged, i.e., $I_k^j = I_k^{j-1}$. In this case, no handover will be triggered.

[4] The transmission delay includes the delay during the handover procedure.

[5] After receiving all the input data of a computation task during a current decision epoch, the UAV starts to process from the subsequent decision epoch since the VMs are created at the beginning of an epoch [9].

[6] To stabilize the training process of the proactive algorithm designed in this work, we choose an exponential function for the definition of an immediate utility, whose value does not dramatically diverge.

as the state value function in a global system state $\boldsymbol{\xi}$ under a joint task scheduling policy $(\pi_k, , \boldsymbol{\pi}_{-k})$ [20].

Due to the shared I/O resource at the UAV and the dynamic nature in networking environment, we formulate the problem of resource awareness among multiple MUs across the decision epochs as a non-cooperative stochastic game, in which the MUs are the players and there are a set $\mathcal{Z}^{|\mathcal{K}|}$ of global system states and a collection of task scheduling policies $\{\pi_k : \forall k \in \mathcal{K}\}$. The aim of each MU k is to device a best-response policy π_k^* that maximizes $V_k(\boldsymbol{\xi}, (\pi_k, \boldsymbol{\pi}_{-k}))$, which can be formally formulated as $\pi_k^* = \arg\max_{\pi_k} V_k(\boldsymbol{\xi}, (\pi_k, \boldsymbol{\pi}_{-k}))$, $\forall \boldsymbol{\xi} \in \mathcal{Z}^{|\mathcal{K}|}$. A NE, which is a best-response task scheduling policy profile $(\pi_k^* : k \in \mathcal{K})$, describes the rational behaviours of the MUs in a stochastic game [19]. In order to operate the NE, a MU has to know the complete global system dynamics, which is prohibited in a non-cooperative networking environment [5]. Define $\mathbb{V}_k(\boldsymbol{\xi}) = V_k(\boldsymbol{\xi}, (\pi_k^*, \boldsymbol{\pi}_{-k}^*))$ as the optimal state-value function.

IV. Proactive DRL with Local Observations

In this section, we shall develop a proactive DRL algorithm to approach the NE task scheduling policy.

A. Approximation from Local Observations

During the competitive interactions with other MUs in the stochastic game, it is challenging for a MU to obtain the global system state information. There still exists the possibility for each MU $k \in \mathcal{K}$ to acquire the side information, which is the partial observation O_k^j, of $\boldsymbol{\xi}_{-k}^j$ during a decision epoch j. In this work, the partial observation of MU k at the beginning of a decision epoch j indicates the remote processing delay at the UAV from the previous epoch $j-1$, namely, $O_k^j = D_{(\mathrm{UAV}),k}^{j-1}$. Therefore, (6) can be approximated by (7), where O_k^1 is the initial partial observation of $\boldsymbol{\xi}_{-k}$. Each MU k then switches to solve the following single-agent MDP,

$$\pi_k^* = \arg\max_{\pi_k} V_k\left(\left(\boldsymbol{\xi}_k, O_k\right), \left(\pi_k, \boldsymbol{\pi}_{-k}\right)\right), \forall (\boldsymbol{\xi}_k, O_k). \quad (8)$$

A dynamic programming approach to (8) based on the value or policy iteration requires complete a priori knowledge of the local state and observation transition statistics [20]. The Q-learning enables each MU k to learn π_k^* in an unknown MEC system. Define

$$\mathbb{Q}_k((\boldsymbol{\xi}_k, O_k), (X_k, F_k)) = (1-\gamma) \cdot u_k(\boldsymbol{\xi}, (X_k, F_k)) + \gamma \cdot \sum_{\boldsymbol{\xi}_k, O_k} \mathbb{P}((\boldsymbol{\xi}_k', O_k') | (\boldsymbol{\xi}_k, O_k), (X_k, F_k)) \cdot \mathbb{V}_k(\boldsymbol{\xi}_k', O_k'), \quad (9)$$

as the Q-function, where X_k and F_k are the decision makings at a current decision epoch, $\boldsymbol{\xi}_k'$ and O_k' are the local state and the partial observation at the subsequent epoch, while $\mathbb{V}_k(\boldsymbol{\xi}_k', O_k') = V_k((\boldsymbol{\xi}_k, O_k), (\pi_k^*, \boldsymbol{\pi}_{-k}^*))$. In turn, $\mathbb{V}_k(\boldsymbol{\xi}_k, O_k)$ can be straightforwardly obtained from

$$\mathbb{V}_k(\boldsymbol{\xi}_k, O_k) = \max_{X_k, F_k} \mathbb{Q}_k((\boldsymbol{\xi}_k, O_k), (X_k, F_k)). \quad (10)$$

By substituting (10) back into (9), we get (11), with X_k' and F_k' denoting the local and remote computation task scheduling decisions under $(\boldsymbol{\xi}_k', O_k')$.

During the process of Q-learning, each MU $k \in \mathcal{K}$ in the network first observes $(\boldsymbol{\xi}_k, O_k) = (\boldsymbol{\xi}_k^j, O_k^j)$, $(X_k, F_k) = (X_k^j, F_k^j)$, $u_k(\boldsymbol{\xi}, X_k, F_k)$ at a current decision epoch j as well as $(\boldsymbol{\xi}_k', O_k') = (\boldsymbol{\xi}_k^{j+1}, O_k^{j+1})$ at the next epoch $j+1$, and then updates the Q-function iteratively as in (12), where $\alpha^j \in [0, 1)$ is the learning rate. It has been well established that if: 1) the global system state transition probability under $(\pi_k^*, \boldsymbol{\pi}_{-k}^*)$ is time-invariant; 2) $\sum_{j=1}^{\infty} \alpha^j$ is infinite and $\sum_{j=1}^{\infty} (\alpha^j)^2$ is finite; and 3) all $((\boldsymbol{\xi}_k, O_k), (X_k, F_k))$-pairs are visited infinitely often, the learning process converges towards π_k^* [20].

B. Proactive DRL for NE Control Policy

We can easily find that for the system model being investigated in this paper, the joint space of local states and partial observations faced by each MU is extremely huge. The tabular nature in representing the Q-function values makes the Q-learning impractical. Inspired by the widespread success of a deep neural network [22], we adopt a double deep Q-network (DQN) to model the Q-function of a MU [23]. However, the accuracy of (8), which is based on partial observations of other MUs in the MEC system, can be, in general, arbitrarily bad. In order to overcome such a challenge from partial observability, we propose a slight modification to the DQN architecture. That is, we replace the first fully-connected layer of the DQN with a long short-term memory (LSTM) layer [24], resulting in a deep recurrent Q-network (DRQN) [10], [25].

More specifically, for each MU $k \in \mathcal{K}$ in the MEC system, $\mathbb{Q}_k((\boldsymbol{\xi}_k, O_k), (X_k, F_k))$ is replaced by $\mathbb{Q}_k(\mathbf{n}_k, (X_k, F_k); \boldsymbol{\theta}_k)$, where $\boldsymbol{\theta}_k$ contains a vector of parameters associated with the DRQN while $\mathbf{n}_k = \mathbf{n}_k^j$ consists of the N most recent local states and partial observations up to a current decision epoch j, namely,

$$\mathbf{n}_k^j = \left(\left(\boldsymbol{\xi}_k^{j-n+1}, O_k^{j-n+1}\right) : n = N, N-1, \cdots, 1\right). \quad (13)$$

It is worth mentioning that $\mathbf{n}_k$ is taken as an input to the LSTM layer of the DRQN of MU k for a proactive and more precise prediction of the current global system state $\boldsymbol{\xi}$. Eventually, a MU leans the parameters of a DRQN, instead of finding the Q-function according to the rule in (12).

C. Offline Training by Digital Twin

Simply being equipped with an independent DRQN at each MU raises two new technical challenges:

1) the possibly asynchronous training of DRQNs at the MUs constrains the overall system performance; and
2) in practice, the limited computation capability at the mobile device of a MU hinders the feasibility of training a DRQN locally.

As a promising alternative, we set up a digital twin of the MEC system to offline train the DRQNs, the parameters of which can be preloaded to a MU during the network initiation. From the assumptions made in this paper and the definition of an identical utility function structure as in (5), the homogeneous behaviours in all MUs provide an opportunity for the digital twin to train a common DRQN with parameters $\boldsymbol{\theta}$. In other

$$V_k((\boldsymbol{\xi}_k, O_k), (\pi_k, \boldsymbol{\pi}_{-k})) = (1-\gamma) \cdot \mathsf{E}_{(\pi_k, \boldsymbol{\pi}_{-k})}\left[\sum_{j=1}^{\infty}(\gamma)^{j-1} \cdot u_k\left(\boldsymbol{\xi}^j, \left(X_k^j, F_k^j\right)\right) \mid \left(\boldsymbol{\xi}_k^1, O_k^1\right) = (\boldsymbol{\xi}_k, O_k)\right] \tag{7}$$

$$\begin{aligned} \mathbb{Q}_k((\boldsymbol{\xi}_k, O_k), (X_k, F_k)) = {} & (1-\gamma) \cdot u_k(\boldsymbol{\xi}, (X_k, F_k)) \\ & + \gamma \cdot \sum_{\boldsymbol{\xi}_k, O_k} \mathbb{P}((\boldsymbol{\xi}'_k, O'_k)|(\boldsymbol{\xi}_k, O_k), (X_k, F_k)) \cdot \max_{X'_k, F'_k} \mathbb{Q}_k((\boldsymbol{\xi}'_k, O'_k), (X'_k, F'_k)) \end{aligned} \tag{11}$$

$$\begin{aligned} \mathbb{Q}_k^{j+1}((\boldsymbol{\xi}_k, O_k), (X_k, F_k)) = {} & \left(1-\alpha^j\right) \cdot \mathbb{Q}_k^j((\boldsymbol{\xi}_k, O_k), (X_k, F_k)) \\ & + \alpha^j \cdot \left((1-\gamma) \cdot u_k(\boldsymbol{\xi}, (X_k, F_k)) + \gamma \cdot \max_{X'_k, F'_k} \mathbb{Q}_k^{j+1}((\boldsymbol{\xi}_k, O_k), (X'_k, F'_k))\right) \end{aligned} \tag{12}$$

words, we derive for each MU $k \in \mathcal{K}$, $\mathbb{Q}_k(\mathbf{n}_k, (X_k, F_k); \boldsymbol{\theta}_k) = \mathbb{Q}(\mathbf{n}_k, (X_k, F_k); \boldsymbol{\theta})$.

To implement the DRQN offline training at the digital twin, we maintain a replay memory $\mathcal{M}^j$ to store the most recent M experiences $\{\mathbf{m}^{j-M+1}, \cdots, \mathbf{m}^j\}$ up to the beginning of each decision epoch j, where an experience $\mathbf{m}^{j-m+1}$ ($m \in \{1, \cdots, M\}$) is given as (14). Meanwhile, a pool $\mathcal{N}^j = \{\mathbf{n}_k^j : k \in \mathcal{K}\}$ of N latest local states and partial observations is kept to predict the global system state $\boldsymbol{\xi}^j$ for task scheduling policy evaluation at epoch j. Both $\mathcal{M}^j$ and $\mathcal{N}^j$ are refreshed over the decision epochs. We first randomly sample a mini-batch $\tilde{\mathcal{M}}^j = \{\breve{\mathcal{M}}^{j_1}, \cdots, \breve{\mathcal{M}}^{j_{\tilde{M}}}\}$ of size $\tilde{M}$ from $\mathcal{M}^j$, where each $\breve{\mathcal{M}}^{j_{\tilde{m}}} \not\subseteq \mathcal{M}^j$ ($\tilde{m} \in \{1, \cdots, \tilde{M}\}$) is given by (15). Then the set $\boldsymbol{\theta}^j$ of parameters at epoch j is updated by minimizing the accumulative loss function, which is defined as in (16), where $\boldsymbol{\theta}_-^j$ is the set of parameters of the target DRQN at a certain previous decision epoch before epoch j.

V. Numerical Experiments

In order to quantify the performance gain from the proposed proactive DRL scheme in a UAV-assisted MEC system, numerical experiments based on TensorFlow [21] are conducted. For experimental purpose, we build up a terrestrial MEC system, which is with $B = 4$ BSs in a 0.4×0.4 Km^2 square area. The BSs are placed at equal distance apart, and the square area is divided into $|\mathcal{L}| = 1600$ locations with each representing a small area of 10×10 m^2. The channel model in [5] and the LOS model in [26] are assumed, respectively, for $G_{b,k}^j$ and $G_{(\mathrm{UAV}),k}^j$, $\forall k \in \mathcal{K}$, $\forall b \in \mathcal{B}$ and $\forall j$. We use the mobility configurations as in [15] for the MUs and the UAV. Regarding the DRQN, we design two fully connected layers after the LSTM layer with each of the three layers containing 32 neurons. ReLU is selected as the activation function [27] and Adam as the optimizer [28]. Other parameter values are listed in Table I.

For the performance comparisons, we design the following four baseline schemes as well.

1) *Local Computation* – Each MU processes locally all arriving computation tasks.
2) *Cloud Execution* – All arriving computation tasks at the MUs are offloaded to the MEC cloud for execution via the BS with the best channel gain.
3) *UAV Execution* – All queued computation tasks from the MUs are processed by the VMs at the UAV.

TABLE I
Parameter values in experiments.

Parameter	Value	Parameter	Value
μ	500 Kbits	ϑ	1300
H	100 meters	W	1 MHz
σ^2	-174 dBm/Hz	δ	10^{-2} second
P_k	3 Watt, $\forall k$	η	3
ρ	2 GHz	φ	0.1
ζ	10^{-3} second	C_0	$2 \cdot 10^7$ bits/second
τ	$2.5 \cdot 10^{-28}$	N	50
M	5000	$\breve{M}$	200

4) *Greedy Processing* – Each MU schedules the local task computation or offloads the computation to the UAV or the MEC cloud whenever possible.

In the experiments, the priority is to demonstrate the average utility performance per MU across the decision epochs from the proposed proactive DRL scheme and the four baselines under various computation task arrival probabilities. We assume $|\mathcal{K}| = 12$ MUs in the MEC system. The results are depicted in Fig. 2. It can be observed from the curves that the average utility performance from the proposed scheme, the local computation, the cloud execution, the UAV execution and the greedy processing deceases as the computation task arrival probability λ increases, which is in accordance with our intuition lying in the surge of per-MU task queue length. Due to the LOS wireless transmissions between the MUs and the UAV, the UAV execution scheme achieves better average utility performance than the cloud execution scheme. As λ increases, more task computations are offloaded for UAV execution under the greedy processing scheme to avoid the possible handover delay, though the cloud execution scheme outperforms the local computation scheme. Among the four baselines, the greedy processing scheme exhibits the best performance under large values of λ. Last but not least, the results clearly show that the proposed scheme provides a significant performance gain, compared with the four baselines.

VI. Conclusions

In this work, our focus is to study the design of a stochastic local and remote computation scheduling policy for each MU in a UAV-assisted MEC system, which takes into account the system dynamics originated from the UAV and the MU mobilities as well as the time-varying computation task arrivals. The

$$\mathbf{m}^{j-m+1} = \left(\left(\left(\boldsymbol{\xi}_k^{j-m}, O_k^{j-m}\right), \left(X_k^{j-m}, F_k^{j-m}\right), u_k\left(\boldsymbol{\xi}^{j-m}, \left(X_k^{j-m}, F_k^{j-m}\right)\right), \left(\boldsymbol{\xi}_k^{j-m+1}, O_k^{j-m+1}\right)\right) : k \in \mathcal{K}\right) \tag{14}$$

$$\breve{\mathcal{M}}^{j_{\tilde{m}}} = \left\{\left(\mathbf{n}_k^{j_{\tilde{m}}}, \left(X_k^{j_{\tilde{m}}}, F_k^{j_{\tilde{m}}}\right), u_k\left(\boldsymbol{\xi}^{j_{\tilde{m}}}, \left(X_k^{j_{\tilde{m}}}, F_k^{j_{\tilde{m}}}\right)\right), \mathbf{n}_k^{j_{\tilde{m}}+1}\right) : k \in \mathcal{K}\right\} \tag{15}$$

$$\begin{aligned} \mathsf{LOSS}\left(\boldsymbol{\theta}^j\right) = \mathsf{E}_{\left\{\breve{\mathcal{M}}^{j_{\tilde{m}}} \in \tilde{\mathcal{M}}^j\right\}}\Bigg[\Bigg(\sum_{k\in\mathcal{K}}\Bigg(&(1-\gamma)\cdot u_k\left(\boldsymbol{\xi}^{j_{\tilde{m}}}, \left(X_k^{j_{\tilde{m}}}, F_k^{j_{\tilde{m}}}\right)\right) + \\ &\gamma \cdot \mathbb{Q}\left(\mathbf{n}_k^{j_{\tilde{m}}+1}, \underset{X_k^{j_{\tilde{m}}+1}, F_k^{j_{\tilde{m}}+1}}{\arg\max}\, \mathbb{Q}\left(\mathbf{n}_k^{j_{\tilde{m}}+1}, \left(X_k^{j_{\tilde{m}}+1}, F_k^{j_{\tilde{m}}+1}\right); \boldsymbol{\theta}^j\right); \boldsymbol{\theta}_-^j\right) - \mathbb{Q}\left(\mathbf{n}_k^{j_{\tilde{m}}}, \left(X_k^{j_{\tilde{m}}}, F_k^{j_{\tilde{m}}}\right); \boldsymbol{\theta}^j\right)\Bigg)\Bigg)^2\Bigg] \end{aligned} \tag{16}$$

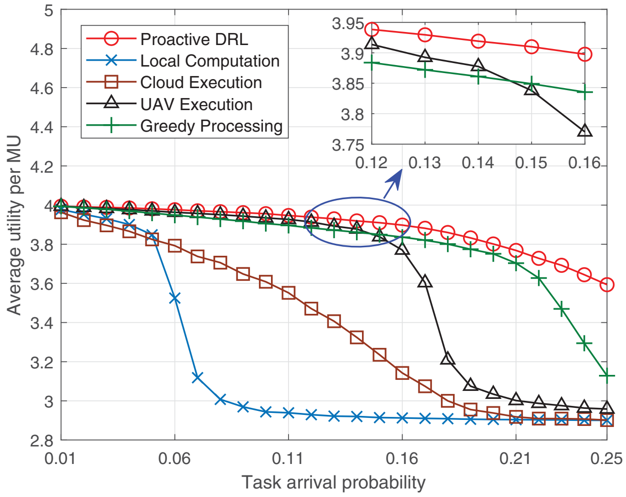

Fig. 2. Average utility performance per MU across the decision epochs versus computation task arrival probability λ.

non-cooperative interactions among the MUs across the decision epochs are formulated as a stochastic game. To approach the NE, we derive a proactive DRL scheme, with which each MU schedules local and remote computations using only the local information. The homogeneity in the behaviours of MUs facilitates the use of a digital twin to offline train the proposed scheme. From numerical experiments, we find that compared with the four baselines, the proposed proactive DRL scheme achieves the best average utility performance.

REFERENCES

[1] Y. Mao *et al.*, “A survey on mobile edge computing: The communication perspective,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Q4 2017.

[2] F. Wang *et al.*, “Joint offloading and computing optimization in wireless powered mobile-edge computing systems,” *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[3] C.-F. Liu, M. Bennis, and H. V. Poor, “Latency and reliability-aware task offloading and resource allocation for mobile edge computing,” in *Proc. IEEE GLOBECOM WKSHP*, Singapore, Dec. 2017.

[4] X. Chen *et al.*, “Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning,” *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[5] X. Chen *et al.*, “Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2377–2392, Oct. 2019.

[6] M. Mozaffari *et al.*, “A tutorial on UAVs for wireless networks: Applications, challenges, and open problems,” *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, Q3 2019.

[7] X. Hu *et al.*, “UAV-assisted relaying and edge computing: Scheduling and trajectory optimization,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.

[8] F. Zhou *et al.*, “Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[9] Z. Liang *et al.*, “Multiuser computation offloading and downloading for edge computing with virtualization,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4298–4311, Sep. 2019.

[10] X. Chen *et al.*, “Age of information-aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective,” 2019. Available: https://arxiv.org/pdf/1908.02047.pdf

[11] O. Naparstek and K. Cohen, “Deep multi-user reinforcement learning for distributed dynamic spectrum access,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[12] R. Dong *et al.*, “Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4692–4707, Oct. 2019.

[13] X. Liu *et al.*, “Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, Aug. 2019.

[14] Y. Wan *et al.*, “A smooth-turn mobility model for airborne networks,” *IEEE Trans. Veh. Technol.*, vol. 62, no. 7, pp. 3359–3370, Sep. 2013.

[15] X. Xi *et al.*, “Efficient and fair network selection for integrated cellular and drone-cell networks,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 923–937, Jan. 2019.

[16] T. D. Burd and R. W. Brodersen, “Processor design for portable systems,” *J. VLSI Signal Process. Syst.*, vol. 13, no. 2–3, pp. 203–221, Aug. 1996.

[17] X. Chen *et al.*, “Efficient multi-user computation offloading for mobile-edge cloud computing,” *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[18] D. Adelman and A. J. Mersereau, “Relaxations of weakly coupled stochastic dynamic programs,” *Oper. Res.*, vol. 56, no. 3, pp. 712–727, Jan. 2008.

[19] A. M. Fink, “Equilibrium in a stochastic n-person game,” *J. Sci. Hiroshima Univ. Ser. A-I*, vol. 28, pp. 89–93, 1964.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[21] M. Abadi *et al.*, “Tensorflow: A system for large-scale machine learning,” in *Proc. OSDI*, Savannah, GA, Nov. 2016.

[22] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[23] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. AAAI*, Phoenix, AZ, Feb. 2016.

[24] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 9, pp. 1735–1780, Nov. 1997.

[25] M. Hausknecht and P. Stone, “Deep recurrent Q-learning for partially observable MDPs,” in *Proc. AAAI*, Austin, TX, Jan. 2015.

[26] Y. Zeng, R. Zhang, and T. J. Lim, “Throughput maximization for UAV-enabled mobile relaying systems,” *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4983–4996, Dec. 2016.

[27] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proc. ICML*, Haifa, Israel, Jun. 2010.

[28] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. ICLR*, San Diego, CA, May 2015.