Task Management for Cooperative Mobile Edge Computing

Li-Tse Hsieh, Hang Liu
The Catholic University of America
Washington, DC, USA

Yang Guo
National Institute of Standards and Technology
Gaithersburg, MD, USA

Robert Gazda
InterDigital Communications, Inc.
Conshohocken, PA, USA

2020 IEEE/ACM Symposium on Edge Computing (SEC) | 978-1-7281-5943-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/SEC50012.2020.00051

***Abstract*—This paper investigates the task management for cooperative mobile edge computing (MEC), where a set of geographically distributed heterogeneous edge nodes not only cooperate with remote cloud data centers but also help each other to jointly process tasks and support real-time IoT applications at the edge of the network. Especially, we address the challenges in optimizing assignment of the tasks to the nodes under dynamic network environments when the task arrivals, node computing capabilities, and network states are non-stationary and unknown a priori. We propose a novel stochastic framework to model the interactions of the involved entities, including the edge-to-edge horizontal cooperation and the edge-to-cloud vertical cooperation. The task assignment problem is formulated and the algorithm is developed based on online reinforcement learning to optimize the performance for task processing while capturing various dynamics and heterogeneities of node computing capabilities and network conditions with no requirement for prior knowledge of them. Further, by leveraging the structure of the underlying problem, a post-decision state is introduced and a function decomposition technique is proposed, which are incorporated with reinforcement learning to reduce the search space and computation complexity. The evaluation results demonstrate that the proposed online learning-based scheme outperforms the state-of-the-art benchmark algorithms.**

Keywords-mobile edge computing (MEC); task assignment; stochastic optimization; reinforcement learning; decomposition

I. INTRODUCTION

The convergence of communication technologies, information processing, embedded systems, and automation has enabled rapid growth of the Internet of Things (IoT). Various things or objects such as sensors, actuators, and smart devices are connected to the Internet to provide new services such as smart cities, intelligent transportation, and industrial control. These emerging applications often involve performing intensive computations on sensor data, e.g. image/video in real time, aiming to realize fast interactions with the surrounding physical world. Mobile edge computing (MEC) has been advocated to support real-time IoT applications. Edge nodes with computing, storage and communication capabilities are co-located or integrated with base stations (BSs), routers, and gateways in the mobile radio access network (RAN) to execute sensor data processing tasks, such as image recognition and object detection, near the data sources at the edge of the network. Compared to the traditional cloud-based solutions, MEC can reduce data transfer time and conserve communication bandwidth by not shipping large volumes of data collected from many sensors to a centralized data center over the Internet, while providing real-time local context-aware services required by emerging IoT applications.

In contrast to centralized cloud data centers, MEC edge nodes are deployed at geographically distributed locations in a RAN, and user requests for computational tasks may arrive at any MEC edge node, instead of a gateway or master node. Individually, edge nodes have limited and heterogeneous computing resources as well as dynamic network conditions. The tasks may be queued at an edge node due to its limited processing capability and even dropped due to the node's bounded buffer. In addition, the workload received by edge nodes exhibits temporal and spatial fluctuations due to the bursty nature of IoT applications and mobility. If edge nodes can forward the unprocessed tasks to nearby edge nodes and/or remote cloud data centers for execution, the overall processing capability will be increased. The horizontal cooperation among edge nodes as well as the vertical cooperation between edge nodes and remote cloud for jointly processing computational tasks can balance the workload and reduce service latency. However, there are non-trivial challenges to manage the MEC services and assign the tasks to be processed at different nodes in a distributed and dynamic MEC network to achieve the optimal system performance: a) both computing resource availability at a node and network communication delay between the nodes should be taken into consideration to make the best task assignment decision for forwarding tasks from one node to another. b) The task arrivals, available computing capabilities at edge nodes, and network delays are time-varying and unknown a priori in many MEC scenarios.

Most research efforts have focused on the problem of offloading tasks from mobile devices to edge nodes [1], [2] or the vertical cooperation in which MEC edge nodes help cloud data centers process delay-sensitive tasks for improved quality of service (QoS) [3], [4]. There are less attentions to investigate the horizontal cooperation among MEC edge nodes for joint task processing. Recently, the authors in [5] proposed an offloading scheme that allows an edge node to forward its tasks to other edge nodes for processing to balance the workload. However, they assume that users submit their tasks to edge nodes at a constant rate and the task arrival rate at an edge node is known. The queuing delay at an edge node and the network delay between the edge nodes are also deterministic and can be known in advance. These

978-1-7281-5943-0/20/$31.00 ©2020 IEEE
DOI 10.1109/SEC50012.2020.00051

assumptions are too idealized for real deployment scenarios. Furthermore, their task assignment algorithm is based on classical convex optimization methods given a static MEC environment, which fails to characterize system dynamics and impacts the performance.

In this paper, we investigate the task assignment and management for cooperative mobile edge computing services under time-varying task arrivals, node computing capabilities, and network states. We cast the task assignment as a dynamic and stochastic optimization problem and develop an online reinforcement learning algorithm to fully explore the synergy among the MEC entities and achieve optimal QoS performance with no assumption on prior knowledge of the underlying network dynamics. Specifically, we propose a novel stochastic framework to model the horizontal cooperation of edge nodes as well as the vertical cooperation between edge nodes and cloud data centers, and capture various dynamics and heterogeneity of node computation capabilities and MEC network conditions. The task assignment problem is formulated as a Markov decision process (MDP). The optimization algorithm is then developed based on online reinforcement learning. In order to reduce the computational complexity and to improve the learning algorithm efficiency, we propose post-decision state estimation and function decomposition techniques by leveraging structure of the underlying problem. Numerical results show that our proposed approach improves the MEC network performance, compared to the existing algorithms. To the best knowledge of the authors, this is the first work to solve the task assignment optimization problem with edge-to-edge horizontal cooperation and edge-to-cloud vertical cooperation under stochastic and dynamic MEC network environments by employing a machine learning-based approach.

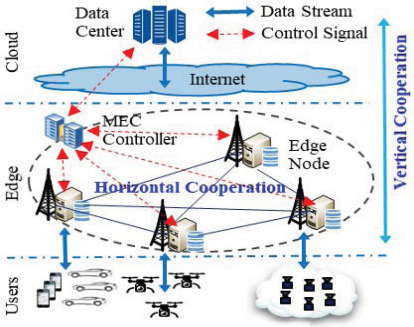

Figure 1. System model.

II. System Model and Problem Formulation

A. System Model

In this paper, we consider a software-defined MEC network with a centralized control plane and a distributed data plane [6]. Software-defined networks (SDNs) have attracted a lot of interest from network service providers because they can be flexibly controlled and programmed. As shown in Fig. 1, a MEC network consists of geographically distributed edge nodes deployed in a RAN covering a certain area. The edge nodes are equipped with computing resources and co-located or integrated with base stations or WiFi access points. They connect to a cloud data center through the Internet. We consider the data center as a special node with powerful resources but far from the RAN. Smart devices/sensors connect to nearby MEC edge nodes to submit their computational tasks, e.g. analyzing sensed video data. The MEC nodes (edge nodes and data center) help each other to jointly process the computational tasks. When an edge node receives the tasks from its associated smart devices, it either process them locally, or forward part or all of its unprocessed tasks to other edge nodes or to the cloud data center for processing to optimize the QoS, which is based on the task assignment decision. In the SDN-based MEC network, a control plane connects the edge nodes to a software-defined programmable MEC controller that makes the task assignment decisions by taking into consideration the network and workload conditions. The MEC controller resides in the RAN and could be one of the edge nodes with dedicated control plane connectivity, thus the control latency is minimal.

Consider a MEC network that consists of N edge nodes, labeled as $\mathcal{N} = \{1, 2, \ldots, N\}$ and a remote cloud data center modeled as a special node n_c. Note that it can be easily extended to multiple data centers. We assume that the system operates over discrete scheduling slots of equal time duration. The values of a two-dimensional task assignment matrix $\boldsymbol{\Phi}^t = \{\phi_{n,j}^t : n, j \in \mathcal{N} \cup n_c\}$ are decided at the beginning of each time slot t, where $\phi_{n,j}^t$ specifies the number of tasks that edge node n will send to edge node j or cloud data center n_c for processing in slot t, and $\phi_{n,n}^t$ is the number of tasks that edge node n will buffer for processing by itself. $\boldsymbol{\phi}_n^t = \{\phi_{n,j}^t, \phi_{j,n}^t : j \in \mathcal{N} \cup n_c\}$ represents the task assignment vector regarding edge node n. We assume that the data center n_c will process all the received tasks by itself, not forwarding them to the edge nodes, i.e. $\phi_{n_c,j}^t = 0, j \in \mathcal{N}$.

B. Problem Formulation

We first formulate the problem of stochastic task assignment optimization and then discuss the approaches to solve the optimization problem. Let A_n^t be the number of the new tasks randomly arrived at edge node n, $n \in \mathcal{N}$ from its associated devices in time slot t, and $\boldsymbol{A^t} = \{A_n^t : n \in \mathcal{N}\}$. The distribution of A_n^t is not known beforehand. Q_n^t represents the task queue length of node n at the beginning of time slot t. Let s_n^t be the task processing capability of node n in slot t, which is defined as the maximal number of tasks that node n can serve in slot t. We assume that s_n^t varies in time and is also unknown a priori. The queue evolution of node n can be written as,

$$Q_n^{t+1} = \begin{cases} 0, & if\ s_n^t \geq Q_n^t + \mathrm{A}_n^t + \Sigma_{i \in e_n} \phi_{i,n}^t - \Sigma_{i \in e_n} \phi_{n,i}^t \\ \min\{Q_n^t + \mathrm{A}_n^t + \Sigma_{i \in e_n} \phi_{i,n}^t - \Sigma_{i \in e_n} \phi_{n,i}^t - s_n^t, Q_n^{(max)}\}, & \text{otherwise} \end{cases} \quad (1)$$

where $Q_n^{(max)}$ is the maximum queue buffer size at node n. An edge node may forward some of its tasks to other nodes for processing, or offer help to process the tasks from other nodes. $\Sigma_{i \in e_n} \phi_{n,i}^t$ where $e_n = \{\mathcal{N} \cup n_c\} \backslash \{n\}$ is the number of tasks that edge node n offloads to other nodes, and $\Sigma_{i \in e_n} \phi_{i,n}^t$ is the

number of tasks that edge node *n* receives from other nodes in slot *t*.

The local state of a node is characterized by its task queue size, its task processing capability, and its network delay to other nodes. For a node n, $n \in \mathcal{N} \cup n_c$, at the beginning of time slot t, we measure its local state as $\boldsymbol{\chi}_n^t = (Q_n^t, s_{n,}^t \boldsymbol{c}_n^t)$ where $\boldsymbol{c}_n^t = \{c_{n,j}^t, c_{j,n}^t : j \in \mathcal{N} \cup n_c\}$ with $c_{n,j}^t$ being the network delay for shipping a task from node n to node j, $c_{j,n}^t$ being to the network delay for shipping a task from node j to node n, and $c_{n,n}^t = 0$. As the network delay between two nodes is related to the transmission distance (the number of hops along the path between the two nodes), traffic conditions in the network, and many other unpredicted factors, it varies in time and its distribution is unknown a priori as well. At the beginning of each scheduling time slot t, the global MEC network state is represented $\boldsymbol{\chi}^t = \{\boldsymbol{\chi}_n^t : n \in \mathcal{N} \cup n_c\} = (\boldsymbol{Q}^t, \boldsymbol{S}^t, \boldsymbol{C}^t) \in X$, where $\boldsymbol{Q}^t = \{Q_n^t : n \in \mathcal{N} \cup n_c\}$, $\boldsymbol{S}^t = \{s_n^t : n \in \mathcal{N} \cup n_c\}$, and $\boldsymbol{C}^t = \{\boldsymbol{c}_n^t : n \in \mathcal{N} \cup n_c\}$. X represents the whole MEC system state space.

For a given MEC network state $\boldsymbol{\chi}^t$ at the beginning of a time slot t, a task assignment $\boldsymbol{\Phi}^t = \boldsymbol{\Phi}(\boldsymbol{\chi}^t) = [\phi_{n,j}(\boldsymbol{\chi}^t) : n, j \in \mathcal{N} \cup n_c\}$ is made, and the MEC network achieves an instantaneous utility that is related to the QoS. We consider delay-sensitive applications, where the QoS is measured by the task service delay and the task drop rate. The task service delay is defined as the period from the time that a task arrives at an edge node to the time that the task has been served in the unit of scheduling slot duration. For an edge node n, $n \in \mathcal{N}$, its service delay d_n depends on the delay incurred by the queue Q_n if edge node n processes the task by itself, or consists of the network delay $c_{n,j}$ and the queueing delay due to the queue Q_j at the service provider j if a task is sent from node n to node j for processing. The task drop rate o_n is defined as the number of tasks dropped per time slot due to buffer overflow.

The instantaneous MEC network utility under the state $\boldsymbol{\chi}^t$ and task assignment decision $\boldsymbol{\Phi}(\boldsymbol{\chi}^t)$ at time slot t is defined as,

$$U(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t)) = \sum_{n \in \mathcal{N}} [w_d U_n^{(d)}(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t)) + w_o U_n^{(o)}(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t))] \quad (2)$$

where $U_n^{(d)}(.)$ and $U_n^{(o)}(.)$ measure the satisfactions of the service delay and task drop rate, respectively. w_d and w_o are the weight factors indicating the importance of delay and task drop in the utility function of the MEC system, respectively. For an edge node, we consider there is a maximal tolerance threshold, $d^{(\max)}$ for the service delay, i.e. $d_n \leq d^{(\max)}$. Correspondingly, let $o^{(\max)}$ be the maximal tolerance threshold for the task drop rate, i.e. $o_n \leq o^{(\max)}$. In addition, we choose the utility function to be the exponential functions, namely $U_n^{(d)} = \exp(-d_n/d^{(\max)})$ and $U_n^{(o)} = \exp(-o_n/o^{(\max)})$ [7].

Stochastic task arrivals and dynamic MEC system states present challenges and make traditional one-shot deterministic optimization schemes unstable and unable to achieve the optimal network performance on a longer timescale. Therefore, we want to develop a stochastic optimization framework for the cooperative task assignment, which maximizes the expected long-term utility of a MEC system while guaranteeing the service delay and task drop rate are within their respective acceptable thresholds.

The task assignment matrix $\boldsymbol{\Phi}(\boldsymbol{\chi}^t)$ is determined according to the control policy $\boldsymbol{\Phi}$ after observing the network state $\boldsymbol{\chi}^t$ at the beginning of a time slot t. The task assignment policy $\boldsymbol{\Phi}$ then induces a probability distribution over the set of possible MEC network states $\boldsymbol{\chi}^{t+1}$ in the following time slot, and hence a probability distribution over the set of per-slot utility $U(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t))$. For simplicity, we assume that the probability of a state in the subsequent slot depends only on the state attained in the present slot, i.e. the task processing capability of a node and the network delay can be modelled as the finite-state discrete-time Markov chains across the time slots. Given a control policy $\boldsymbol{\Phi}$, the random process $\boldsymbol{\chi}^t$ is thus a controlled Markov chain with the following state transition probability [8], [9],

$$\Pr\{\boldsymbol{\chi}^{t+1} | \boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t)\} = \Pr\{\boldsymbol{Q}^{t+1} | \boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t)\} \Pr\{\boldsymbol{S}^{t+1} | \boldsymbol{S}^t\} \Pr\{\boldsymbol{C}^{t+1} | \boldsymbol{C}^t\} \quad (3)$$

For a controlled Markov chain, the transition probability from a present state $\boldsymbol{\chi}^t$ to the next state $\boldsymbol{\chi}^{t+1}$ depends only on the present state $\boldsymbol{\chi}^t$ and the control policy $\boldsymbol{\Phi}(\boldsymbol{\chi}^t)$ acted on the present state. Taking the discounted expectation with respect to the per-slot utilities $U(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t))$ over a sequence of network states $\boldsymbol{\chi}^t$, we can obtain the discounted expected value of the MEC network utility [8],

$$V(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathrm{E}[\alpha \cdot \sum_{t=1}^{\infty} \gamma^{t-1} U(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t)) | \boldsymbol{\chi}^1], \quad (4)$$

where $\alpha, \gamma \in [0, 1)$ are the parameters. γ is a discount factor that discounts the utility rewards received in the future, and $(\gamma)^{t-1}$ denotes the discount to the (t −1)-th power. $\boldsymbol{\chi}^1$ is the initial network state. $V(\boldsymbol{\chi}, \boldsymbol{\Phi})$ is also termed as the state value function of the MEC network in state $\boldsymbol{\chi}$ under task assignment policy $\boldsymbol{\Phi}$. We let $\alpha = 1 - \gamma$, thus, the expected undiscounted long-term average utility, $\bar{U}(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathrm{E}\left[\lim_{T \to \infty} \frac{1}{T} \cdot \sum_{t=1}^{T} U(\boldsymbol{\chi}^t, \boldsymbol{\Phi}(\boldsymbol{\chi}^t)) \middle| \boldsymbol{\chi}^1\right]$ can be considered as a special case of (4) when γ approaches 1 and $\alpha = (1 - \gamma)$ approaches 0 [9]. On the other hand, if γ is set to be 0, then $V(\boldsymbol{\chi}, \boldsymbol{\Phi}) = U(\boldsymbol{\chi}^1, \boldsymbol{\Phi}(\boldsymbol{\chi}^1))$, that is, only the immediate utility performance is considered. We therefore consider the expected discounted long-term utility performance in (4) as a general QoS indicator in this paper.

The objective is to design an optimal task assignment control policy $\boldsymbol{\Phi}^*$ that maximizes the expected discounted long-term utility performance, that is,

$$\boldsymbol{\Phi}^* = arg \max_{\boldsymbol{\Phi}} (V(\boldsymbol{\chi}, \boldsymbol{\Phi})) \quad (5)$$

$V^*(\boldsymbol{\chi}) = V(\boldsymbol{\chi}, \boldsymbol{\Phi}^*)$ is the optimal state value function. The stochastic task assignment optimization in (5) can be considered as a MDP with the discounted utility criterion since the network states follow a controlled Markov process. The optimal task assignment control policy achieving the maximal state value function can thus be obtained by solving the following Bellman's optimality equation [9], [10],

$$V^*(\boldsymbol{\chi}) = \max_{\boldsymbol{\Phi}}\{(1-\gamma)\,U(\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})) + \gamma\sum_{\boldsymbol{\chi}'}\Pr\{\boldsymbol{\chi}'|\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})\}V^*(\boldsymbol{\chi}')\}, \quad (6)$$

where $\boldsymbol{\chi}' = (\boldsymbol{Q}', \boldsymbol{S}', \boldsymbol{C}')$ is the MEC network state in the subsequent time slot, and $\Pr\{\boldsymbol{\chi}'|\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})\}$ represents the state transition probability that making the task assignment $\boldsymbol{\Phi}(\boldsymbol{\chi})$ in state $\boldsymbol{\chi}$ will produce the next state $\boldsymbol{\chi}'$. $\boldsymbol{Q}' = \{Q'_n : n \in \mathcal{N} \cup n_c\}$, $\boldsymbol{S}' = \{s'_n : n \in \mathcal{N} \cup n_c\}$, and $\boldsymbol{C}' = \{\boldsymbol{c}'_n : n \in \mathcal{N} \cup n_c\}$ are the queue, task processing capability, and network delay states in the subsequent time slot.

Solving (6) is generally a challenging problem. Traditional approaches are based on value iteration, policy iteration, and dynamic programming [11], [12]. However, these methods require full knowledge of the network state transition probabilities and task arrival statistics that cannot be known beforehand for our problem.

III. Problem Simplification and Online Learning Algorithm

In this section, we focus on developing an algorithm to obtain the optimal task assignment policy with no requirement for prior knowledge of the statistical information about network state transitions and task arrivals by employing online reinforcement learning techniques [13], [14]. However, the task assignment optimization problem in (6) is very complex; both the MEC system state space and the control action space are very large as discussed later. To solve it, first, we simplify the problem by introducing a post-decision state and then reduce the number of system states through decomposition.

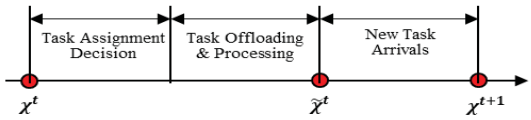

Figure 2. Three phases of a time slot.

Based on the observation that task arrivals are independent of the task assignment policy, we define an intermediate state called post-decision state for each scheduling slot, which is the state after an edge node finishes task offloading to other nodes and local processing. A time slot can be considered consisting of three phases, task assignment decision, task offloading and processing, and new task arrivals as shown in Fig. 2. In phase I, the MEC controller determines the task assignment matrix $\boldsymbol{\Phi}(\boldsymbol{\chi})$ and informs the edge nodes of the task assignment decision. In phase II, an edge node offloads tasks to other nodes or receives tasks from other nodes and processes their tasks based on the task assignment decision. The network state then moves into the post-decision state. The new tasks from the associated devices will arrive at edge nodes in phase III. Note that the three phases and the post-decision state are used to derive the optimal task assignment. In practice, the tasks may arrive at an edge node at any time, and the edge node can process the tasks in its queue and forward the tasks to other nodes during the whole slot time.

At the current scheduling slot, we define the post-decision state as $\tilde{\boldsymbol{\chi}} = (\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{C}})$, where the node processing and network delay states of the post-decision will remain the same as those at the beginning of the time slot, that is, $\tilde{\boldsymbol{S}} = \{\tilde{s}_n : n \in \mathcal{N} \cup n_c\}$ with $\tilde{s}_n = s_n$ and $\tilde{\boldsymbol{C}} = \{\tilde{\boldsymbol{c}}_n : n \in \mathcal{N} \cup n_c\}$ with $\tilde{\boldsymbol{c}}_n = \boldsymbol{c}_n$, respectively, because they are independent of the task assignment decision. The queue state of post-decision is $\tilde{\boldsymbol{Q}} = \{\tilde{Q}_n : n \in \mathcal{N} \cup n_c\}$ with $\tilde{Q}_n = \max\{Q_n + \Sigma_{i\in e_n}\phi_{i,n} - \Sigma_{i\in e_n}\phi_{n,i} - s_n, 0\}$. The probability of MEC network state transition from $\boldsymbol{\chi}$ to $\boldsymbol{\chi}'$ can then be expressed as,

$$\Pr\{\boldsymbol{\chi}'|\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})\} = \Pr\{\boldsymbol{\chi}'|\tilde{\boldsymbol{\chi}}\}\Pr\{\tilde{\boldsymbol{\chi}}|\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})\} = \prod_{n,j\in\mathcal{N}\cup n_c}\Pr\{A_n\}\Pr\{s'_n|s_n\}\Pr\{c'_n|c_n\} \quad (7)$$

where $\Pr\{\tilde{\boldsymbol{\chi}}|\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})\} = 1$ and $A_n = Q'_n - \tilde{Q}_n$. We can control the task assignment decision to ensure that no task drop occurs in the transition to the post-decision state, i.e. the task drop due to buffer overflow may happen only when the new tasks arrive. By introducing the post-decision state, we are able to factor the utility function in (2) into two parts, which correspond to $U_n^{(d)}$ and $U_n^{(o)}$. Then, the optimal state value function satisfying (6) can hence be rewritten by,

$$V^*(\boldsymbol{\chi}) = \max_{\boldsymbol{\Phi}}\{(1-\gamma)\textstyle\sum_{n\in\mathcal{N}} w_d U_n^{(d)}(\boldsymbol{\chi},\boldsymbol{\Phi}(\boldsymbol{\chi})) + \tilde{V}^*(\tilde{\boldsymbol{\chi}})\} \quad (8)$$

where $\tilde{V}^*(\tilde{\boldsymbol{\chi}})$ is the optimal post-decision state value function. that satisfies Bellman's optimality equation,

$$\tilde{V}^*(\tilde{\boldsymbol{\chi}}) = (1-\gamma)\textstyle\sum_{n\in\mathcal{N}} w_0 U_n^{(0)}(\boldsymbol{\chi},\ \boldsymbol{\Phi}^*(\boldsymbol{\chi})) + \gamma\sum_{\boldsymbol{\chi}'}\Pr\{\boldsymbol{\chi}'|\tilde{\boldsymbol{\chi}}\}V^*(\boldsymbol{\chi}') \quad (9)$$

From (8), we find that the optimal state value function can be obtained from the optimal post-decision state value function by performing maximization over all feasible task assignment decisions. The optimal task assignment policy is thus expressed as follows, which should satisfy the maximal delay and task drop constraints.

$$\boldsymbol{\Phi}^* = \operatorname*{argmax}_{\boldsymbol{\Phi}}\{(1-\gamma)\textstyle\sum_{n\in\mathcal{N}} w_d U_n^{(d)}(\boldsymbol{\chi},\Phi(\boldsymbol{\chi})) + \tilde{V}^*(\tilde{\boldsymbol{\chi}})\} \quad \text{s.t. } d_n \le d^{(\max)} \text{ and } o_n \le o^{(\max)} \quad (10)$$

The task arrival statistics and task processing capability of the edge nodes are independent each other. We can then decompose the optimal post-decision state value function [15]. Mathematically, that is

$$\tilde{V}^*(\tilde{\boldsymbol{\chi}}) = \textstyle\sum_{n\in\mathcal{N}} \tilde{V}_n^*(\tilde{Q}_n, \tilde{s}_n, \tilde{\boldsymbol{c}}_n) \quad (11)$$

Given the optimal control policy $\boldsymbol{\Phi}^*$, according to (9) and (11), the post-decision state value function $\tilde{V}_n^*(\tilde{Q}_n, \tilde{s}_n, \tilde{\boldsymbol{c}}_n)$ satisfies,

$$\tilde{V}_n^*(\tilde{Q}_n, \tilde{s}_n, \tilde{\boldsymbol{c}}_n) = (1-\gamma)w_0 U_n^{(0)*}(Q_n, s_n, \boldsymbol{c}_n) + \gamma\textstyle\sum_{A_n, s'_n, c'_n}\Pr\{A_n\}\Pr\{s'_n|s_n\}\Pr\{\boldsymbol{c}'_n|\boldsymbol{c}_n\}V_n^*(Q'_n, s'_n, \boldsymbol{c}'_n) \quad (12)$$

Based on (8) and (11), the optimal state value function of edge node n in the subsequent time slot, $V_n^*(Q'_n, s'_n, \boldsymbol{c}'_n)$ can be expressed as,

$$V_n^*(Q'_n, s'_n, \boldsymbol{c}'_n) = (1-\gamma)w_d U_n^{(d)*}(Q'_n, s'_n, \boldsymbol{c}'_n) + \tilde{V}_n^*(\tilde{Q}'_n, \tilde{s}'_n, \tilde{\boldsymbol{c}}'_n) \quad (13)$$

where $\tilde{Q}_n'$, $\tilde{s}_n'$ and $\tilde{\boldsymbol{c}}_n'$ are the local post-decision queue, processing, and network delay states for node n in the subsequent scheduling slot, respectively.

The linear decomposition of the post-decision state value function proposed above yields two main benefits. First, in order to derive a task assignment policy based on the global MEC network state, $\boldsymbol{\chi} = \{\boldsymbol{\chi}_n : n \in \mathcal{N} \cup n_c\}$ with $\boldsymbol{\chi}_n = (Q_n, s_n, \boldsymbol{c}_n)$ and $\boldsymbol{c}_n = \{c_{n,j}, c_{j,n} : j \in \mathcal{N} \cup n_c\}$, at least $\prod_{n\in\mathcal{N}\cup n_c}\prod_{j\in\mathcal{N}\cup n_c}(|Q_n||s_n||c_{n,j}||c_{j,n}|)$ state values should be kept. Using linear decomposition (11), only $(N+1)|Q_n||s_n|\prod_{j\in\mathcal{N}\cup n_c}(|c_{n,j}||c_{j,n}|)$ values need to be stored, significantly reducing the search space in the task assignment decision making. Second, the problem to solve a complex post-decision Bellman's optimality equation (9) is broken into simpler MDPs. By replacing the post-decision state value function in (10) with (11), we can obtain an optimal task assignment policy $\boldsymbol{\Phi}^*$ under a MEC network state $\boldsymbol{\chi}$.

As discussed before, the number of new task arrivals at the end of a scheduling slot as well as the task processing capability of a node and the states of network delay between the nodes for the next scheduling slot are unknown beforehand. In this case, instead of directly computing the post-decision state value functions in (12), we propose an online reinforcement learning algorithm to learn $\tilde{V}_n^*(\tilde{Q}_n, \tilde{s}_n, \tilde{\boldsymbol{c}}_n), \forall n \in \mathcal{N}$ on the fly. Based on the observations of the network state $\boldsymbol{\chi}_n^t = (Q_n^t, s_n^t \boldsymbol{c}_n^t), \forall n \in \mathcal{N}$, the number of task arrivals $A_n^t, \forall n \in \mathcal{N}$, the decision on the number of tasks locally processed, the number of tasks offloaded to other nodes or received from other nodes, the achieved utility $U_n^{(o)*}(Q_n, s_n, \boldsymbol{c}_n)$ at the current scheduling slot t, and the resulting network state $\boldsymbol{\chi}_n^{t+1} = (Q_n^{t+1}, s_n^{t+1}\boldsymbol{c}_n^{t+1})$ at the next slot $t+1$, the post-decision state value function for node n can be updated by,

$$\tilde{V}_n^{t+1}(\tilde{Q}_n^t, \tilde{s}_n^t, \tilde{\boldsymbol{c}}_n^t) = (1-\varepsilon^t)\tilde{V}_n^t(\tilde{Q}_n^t, \tilde{s}_n^t, \tilde{\boldsymbol{c}}_n^t) + \varepsilon^t[(1-\gamma)w_0 U_n^{(0)}(Q_n^t, s_n^t, \boldsymbol{c}_n^t) + \gamma V_n^t(Q_n^{t+1}, s_n^{t+1}, \boldsymbol{c}_n^{t+1})] \quad (14)$$

where $\varepsilon^t \in [0,1)$ is the learning rate. The task assignment matrix $\boldsymbol{\Phi}^t = [\phi_{n,j}^t : n, j \in \mathcal{N} \cup n_c]$ at scheduling slot t is determined as,

$$\boldsymbol{\Phi}^t = \underset{\boldsymbol{\Phi}}{\operatorname{argmax}}\{\sum_{n\in\mathcal{N}}[(1-\gamma)w_d U_n^{(d)}(Q_n^t, s_n^t, \boldsymbol{c}_n^t) + \tilde{V}_n^t(\tilde{Q}_n^t, \tilde{s}_n^t, \tilde{\boldsymbol{c}}_n^t)]\}$$
$$\text{s.t. } d_n^t \le d^{(\max)} \text{ and } o_n^t \le o^{(\max)} \quad (15)$$

The state value function of node n at slot $t+1$ is evaluated by,

$$V_n^t(Q_n^{t+1}, s_n^{t+1}, \boldsymbol{c}_n^{t+1}) = (1-\gamma)w_d U_n^{(d)}(Q_n^{t+1}, s_n^{t+1}, \boldsymbol{c}_n^{t+1}) + \tilde{V}_n^t(\tilde{Q}_n^{t+1}, \tilde{s}_n^{t+1}, \tilde{\boldsymbol{c}}_n^{t+1}) \quad (16)$$

The online learning algorithm for estimating the optimal post-decision state value function and determining the optimal task assignment policy is summarized in Algorithm 1.

Algorithm 1. Online Learning Algorithm for Optimal Post-Decision State Value Function

1. Initialize the post-decision state value functions $\tilde{V}_n^t(\tilde{\chi}_n^t)$, $\forall \tilde{\chi}_n^t$ and $\forall n \in \mathcal{N}$ for $t = 1$.
2. At the beginning of scheduling slot t, the MEC controller observes the network state, $\boldsymbol{\chi}^t = \{\boldsymbol{\chi}_n^t : n \in \mathcal{N}\}$ with $\boldsymbol{\chi}_n^t = (Q_n^t, s_n^t, \boldsymbol{c}_n^t)$ and determines the task assignment matrix, $\boldsymbol{\Phi}^t = [\boldsymbol{\phi}_n^t : n \in \mathcal{N}]$ according to (15).
3. After offloading and processing the tasks according to the above task assignment decision, the controller observes the post-decision state, $\tilde{\boldsymbol{\chi}}^t = \{\tilde{\boldsymbol{\chi}}_n^t : n \in \mathcal{N}\}$, where $\tilde{\boldsymbol{\chi}}_n^t = (\tilde{Q}_n^t, \tilde{s}_n^t, \tilde{\boldsymbol{c}}_n^t)$ with $\tilde{Q}_n^t = \max\{Q_n^t + \Sigma_{i\in e_n}\phi_{i,n}^t - \Sigma_{i\in e_n}\phi_{n,i}^t - s_n^t, 0\}$, $\tilde{s}_n^t = s_n^t$, and $\tilde{\boldsymbol{c}}_n^t = \boldsymbol{c}_n^t$.
4. With $\boldsymbol{A}^t = \{A_n^t : n \in \mathcal{N}\}$ new tasks arrived at the end of slot t, the network state transits to $\boldsymbol{\chi}^{t+1} = \{\boldsymbol{\chi}_n^{t+1} : n \in \mathcal{N}\}$ where $\boldsymbol{\chi}_n^{t+1} = (\tilde{Q}_n^t + A_n^t, s_n^{t+1}, \boldsymbol{c}_n^{t+1})$ at the following scheduling slot t+1.
5. Calculate $V_n^t(Q_n^{t+1}, s_n^{t+1}, \boldsymbol{c}_n^{t+1})$, $\forall n \in \mathcal{N}$ according to (16) and updates the post-decision state value functions $\tilde{V}_n^{t+1}(\tilde{Q}_n^t, \tilde{s}_n^t, \tilde{\boldsymbol{c}}_n^t)$, $\forall n \in \mathcal{N}$ according to (14).
6. The scheduling slot index is updated by $t \leftarrow t+1$.
7. Repeat from step 2 to 6.

IV. Numerical Results

We provide the evaluation results in this section and compare the performance of our online reinforcement learning scheme with several benchmark schemes including i) no cooperation, i.e. an edge node processes all the tasks it receives from its associated devices by itself; ii) cloud execution, i.e. an edge node offloads all its received tasks to the cloud data center for execution; iii) one-shot deterministic optimization which is similar to the scheme in [5].

We simulated multiple MEC network scenarios with different system parameters. Due to the page limit, we present the results for a typical setting. We assume the slot duration is 30 ms. The task processing capability of an edge node is considered to be an independent Markov chain model with three states {4, 2, 1} tasks per slot. The network delay between two edge nodes is also modeled as a Markov chain with three states, {1, 0.5, 0.2} slots. The cloud data center has powerful computation resources, and the queuing and processing delay in the cloud data center is small enough to be ignored, but forwarding the tasks to the cloud incurs a large network delay, 10 slots, due to a long distance with many hops over the Internet.

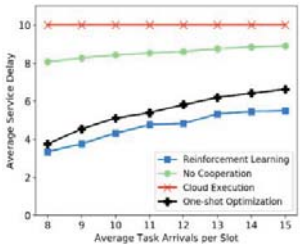

Figure 3. The average task service delay versus the average task arrivals per slot for different algorithms.

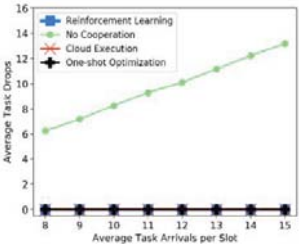

Figure 4. The average number of dropped tasks per slot versus the average task arrivals per slot for different algorithms.

In Fig. 3, we compare the average task service delay for different algorithms with three edge nodes and one cloud data center when the task arrivals follow independent Poisson arrival process and the average number of task arrivals per slot

at an edge node changes. Note that the delay is measured in the unit of the time slot duration. The curves indicate that our proposed online reinforcement learning scheme outperforms all the three benchmark schemes. Compared to the one-shot optimization algorithm (the second best in terms of the service delay), the proposed online learning scheme can capture the dynamic MEC network state transitions and determine the optimal task assignment matrix by taking into consideration the impacts of time-varying stochastic network environments and node computing capabilities to the expected long-term performance in the future. However, the one-shot deterministic optimization algorithm makes shortsighted task assignment decisions and may cause a lot of tasks to be shipped to the cloud data center for processing, which leads to a large network delay and thus a large service delay. In Fig. 4, we present the average number of the tasks dropped per slot for different algorithms. The task drops for the online learning and one-shot optimization algorithms are close to zero because the algorithms minimize the task drops and the edge nodes will forward the tasks to the cloud data center when their buffers become full. For the no cooperation scheme, when the workload is high, an edge node does not have enough resources to process all the tasks so that the service delay increases and the tasks are dropped. For the cloud execution scheme, there is always a large value of network delay to ship the tasks to the cloud data center for processing over the Internet.

V. CONCLUSIONS

In many MEC scenarios, the task arrival statistics, task processing capability at an edge node, and network delay between two nodes are time-varying and unknown beforehand. Therefore, casting the task assignment as a dynamic and stochastic optimization problem is more reasonable and compelling. In this paper, we have proposed and investigated a stochastic framework to model the interactions among various entities of a MEC system, including the edge-to-edge horizontal cooperation and the edge-to-cloud vertical cooperation for jointly processing tasks under dynamic and uncertain network environments. The task assignment optimization problem is formulated as a Markov decision process by taking into consideration the non-stationary computing and network states as well as the interaction and heterogeneity of the involved entities. To solve the problem, we derive an algorithm based on online reinforcement learning, which learns on the fly the optimal task assignment policy without prior knowledge of task arrival and network statistics. Further, considering the structure of the underlying problem, we introduce a post-decision state and a function decomposition technique to reduce the search space, which are combined with reinforcement learning. The evaluation results show that the proposed online learning-based scheme reduces the service delay, compared to the existing schemes that do not consider dynamic changes in traffic and MEC network statistics.

ACKNOWLEDGMENT

Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

REFERENCES

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

[2] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[3] C. Do, N. Tran, C. Pham, M. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *Proc. of the IEEE ICOIN*, pp. 324–329, Siem Reap, Cambodia, Jan. 2015.

[4] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1204–1215, 2017.

[5] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. of IEEE INFOCOM'17*, Atlanta, GA, May 2017.

[6] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, Y. Zhang, "Mobile Edge Cloud System: Architectures, Challenges, and Approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495-2508, Sept. 2018.

[7] X. Chen, Z. Han, H. Zhang, G. Xue, Y. Xiao, and M. Bennis, "Wireless resource scheduling in virtualized radio access networks using stochastic learning*," IEEE Transactions on Mobile Computing*, vol. 17, no. 4, pp. 961-974, 2018.

[8] S. M. Ross, Introduction to stochastic dynamic programming. Academic press, 2014.

[9] R. Howard, Dynamic Programming and Markov Processes. MIT Press, 1960.

[10] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA, 1995.

[11] M. L. Puterman and M. C. Shin, "Modified policy iteration algorithms for discounted Markov decision problems," *Management Science*, vol. 24, no. 11, pp. 1127–1137, 1978.

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

[13] X. Chen, P. Liu, H. Liu, C. Wu, Y. Ji, "Multipath Transmission Scheduling in Millimeter Wave Cloud Radio Access Networks," in *Proceedings of IEEE ICC'18*, Kansas City, MO, May 2018.

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

[15] J. N. Tsitsiklis and B. van Roy, "Feature-based methods for large scale dynamic programming," *Mach. Learn.*, vol. 22, no. 1-3, pp. 59 - 94, Jan. 1996.