

# Online Distributed Job Dispatching with Outdated and Partially-Observable Information

Yuncong Hong<sup>†‡§</sup>, Bojie Lv<sup>†§</sup>, Rui Wang<sup>†§</sup>, Haisheng Tan<sup>\*§</sup>, Zhenhua Han<sup>†</sup>, Hao Zhou<sup>\*</sup>, Francis C.M. Lau<sup>‡</sup>

<sup>\*</sup> LINKE Lab, University of Science and Technology of China, Hefei, China

<sup>†</sup> Department of Electrical and Electronic Engineering, Southern University of Science and Technology, Shenzhen, China

<sup>‡</sup> Department of Computer Science, The University of Hong Kong, Hong Kong, China

<sup>§</sup> Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China

**Abstract**—In this paper, we investigate online distributed job dispatching in an edge computing system residing in a Metropolitan Area Network (MAN). Specifically, job dispatchers are implemented on access points (APs) which collect jobs from mobile users and distribute each job to a server at the edge or the cloud. A signaling mechanism with periodic broadcast is introduced to facilitate cooperation among APs. The transmission latency is non-negligible in MAN, which leads to outdated information sharing among APs. Moreover, the fully-observed system state is discouraged as reception of all broadcast is time consuming. Therefore, we formulate the distributed optimization of job dispatching strategies among the APs as a Markov decision process with partial and outdated system state, *i.e.*, partially observable Markov Decision Process (POMDP). The conventional solution for POMDP is impractical due to huge time complexity. We propose a novel low-complexity solution framework for distributed job dispatching, based on which the optimization of job dispatching policy can be decoupled via an alternative policy iteration algorithm, so that the distributed policy iteration of each AP can be made according to partial and outdated observation. A theoretical performance lower bound is proved for our approximate MDP solution. Furthermore, we conduct extensive simulations based on the Google Cluster trace. The evaluation results show that our policy can achieve as high as 20.67% reduction in average job response time compared with heuristic baselines, and our algorithm consistently performs well under various parameter settings.

## I. INTRODUCTION

Edge computing is a promising solution for increasing computation-intensive and energy-hungry applications on mobile devices. Large amount of mobile devices can connect to the access points (APs) which function as gateways to aggregate and dispatch jobs to the edge servers [1]. The edge servers are deployed in closer proximity to APs than cloud infrastructures, which alleviate the communication overhead and enable the computation of time-sensitive jobs. However, the edge servers are usually deployed with limited computation resources. The establishment of efficient cooperation among edge servers is one of the major design challenges, given the signaling overhead and latency of distributed information exchange and decision making.

We consider an edge computing system with multiple APs and edge servers residing in the Metropolitan Area Network (MAN) as illustrated in Fig.1. The APs collect jobs offloaded from the mobile users in its service area and make dispatching decision for each job. According to the MAN performance

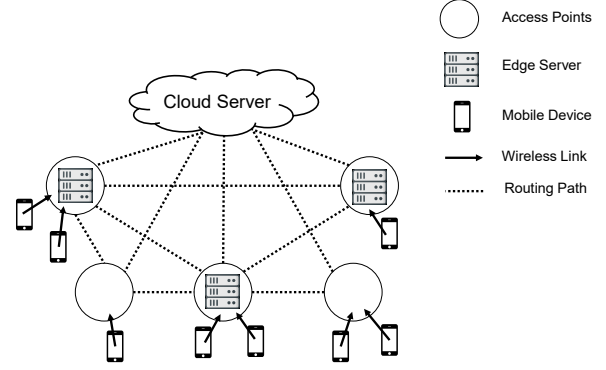


Fig. 1. The Illustration of System Model

analysis in [2], the data transmission latency among APs and edge servers varies a lot with respect to different hours of day and devices' locations in a MAN. In addition, each AP also suffers from signaling latency, which is the time consumed for each AP to collect system state information under some signaling mechanism. There has been a number of existing literature considering random transmission latency of job delivery in edge computing networks (e.g., [3]–[7]). However, there are only a few works considering signaling latency of cooperation among distributed job dispatchers [8], [9]. In fact, it is full of challenge to consider latency in both job delivery and signaling. The cooperation of distributed dispatchers suffers from significant unpredictable signaling and transmission overhead. Therefore, aggregating the global system state information at each AP is not practical, and a centralized dispatcher design is discouraged. Moreover, the latency will also lead to outdated information at each dispatcher and information inconsistency among different dispatchers, which may introduce ineliminable estimation error on the number of jobs in the system and thus discourages the cooperative dispatcher design.

In this paper, we would like to shed some lights on the above challenging distributed dispatcher design via proposing the POMDP (partially observable Markov decision process) problem formulation and a novel low-complexity approximate MDP solution framework. Specifically, we consider a practical scenario where the signaling latency among APs and edge servers as well as job uploading latency from APs to edge

servers are assumed to be random, and each AP can only receive the broadcast system state information from part of the APs with random latency (i.e., the global system state is not available at the APs). Our main contributions in this new optimization scenario are summarized as follows.

- We propose a novel low-complexity distributed solution framework for the dispatcher design at APs. By leveraging partial observation at each dispatcher, we derive the expression of approximate value function under MDP framework and decouple the optimization problem onto each AP. Thus, the complicated POMDP solution is avoided. To our best knowledge, this is the first work to address the cooperative distributed multi-agent optimization problem with outdated and partial information under MDP framework.
- We derive an analytical cost lower bound for the proposed distributed dispatching policy in the above low-complexity solution framework. In the conventional approximate MDP methods, the solution is usually evaluated via numerical method which are hard to obtain analytical performance bound.
- We conduct extensive simulations based on the Google Cluster trace, compared with three heuristic benchmarks. The evaluation results show that our proposed job dispatching policy can achieve 20.67% reduction in average job response time, and our algorithm consistently performs well under various parameter settings of signaling latency, job arrival intensity and job processing time.

**Paper Organization:** The remainder of this paper is organized as follows. In Section II, the related works are elaborated. In Section III, we illustrate the system model and the signaling model with random latency. In Section IV, we formulate the global optimization of dispatching decisions at all APs as an POMDP. In Section V, we introduce the novel low-complexity distributed solution framework for the above POMDP. The numerical analysis of the proposed solution is provided in Section VI, and the conclusion is drawn in Section VII.

## II. RELATED WORK

There have been a number of works considering the centralized job dispatching with updated and complete knowledge on the system states of edge computing systems. For example, in order to minimize the average job response time in the worst case, the authors in [10] designed an online algorithm for job dispatching in edge computing systems with fixed uploading latency. In the scenario that APs and edge servers are connected via software defined network (SDN), the authors in [5] proposed a heuristic algorithm to dispatch the jobs to the closest edge servers according to their locations. Considering random jobs arrival and job offloading to a single edge server, the authors in [11], [12] formulate the offloading problem as an infinite-horizon Markov decision process (MDP). In the above works, a centralized dispatcher with complete and updated knowledge of the system states was assumed in the edge computing systems, which might be impractical.

Hence, there are also some works considering the distributed job dispatching in edge computing systems. For example, in order to minimize a weighted sum of total energy consumption and uploading latency, the authors in [13] proposed a distributed job dispatching algorithm based on game theory to achieve the Nash equilibrium. Considering job migration at edge servers, the authors in [14] optimized the edge computing performance in a distributed manner with limited energy resources via a congestion game framework. In the scenario that APs cooperatively dispatch jobs with multiple edge servers, the authors in [15] proposed a novel approximate MDP solution framework to alleviate the algorithm complexity and minimize the average job response time. However, in the above works, the latency of information exchange among APs and edge servers is ignored. In fact, due to the complicated network traffic, this latency might be significant, and the staleness of system state information at the dispatcher of a edge computing systems should be considered.

The staleness of information sharing among APs and edge servers may degrade the performance of the job dispatching algorithm in edge computing systems. To the best of our knowledge, there are very limited works investigating this issue. For example, the authors in [8] proposed a randomized policy via Lyapunov optimization approach to stabilize the queues in a MEC system with multiple IoT devices offloading jobs to one edge server, where signaling latency is considered. In [9], the above approach was applied to the scenario that mobile devices offload jobs to each other via D2D link. In the above two works, there is one centralized dispatcher in the system and the objective is to stabilize the transmission queues. Hence, the existence of signaling latency may not raise significant challenge to the algorithm designs.

However, the design of distributed dispatchers with signaling latency could be more challenging. For example, the signaling latency at distributed dispatchers could be different, and the synchronization of their dispatching decisions become infeasible. Furthermore, taking the signaling overhead in consideration, it is of more practical significance favor for the distributed dispatchers to make decisions based on locally observed system state information, instead of global system state information. To our best knowledge, there is no appropriate optimization framework for the distributed dispatcher design with both signaling latency and partially observable system state information to date.

## III. SYSTEM MODEL

### A. Network Model

We consider an edge computing system with  $K$  Access Points (APs),  $M$  edge servers, and 1 cloud server as illustrated in Fig.1. The sets of APs and processing servers are denoted as  $\mathcal{K} \triangleq \{1, \dots, K\}$  and  $\mathcal{M} \triangleq \{0, \dots, M\}$ , respectively. Specifically, we have *processing server* denote both edge servers and the cloud server for job processing and the 0-th processing server denote the cloud server. One edge server is always collocated with one AP, i.e., the edge server is deployed at the same place with the access point. Each AP

collects the jobs from the mobile users within its coverage, and makes dispatching action on the processing servers for each job. Furthermore, it is assumed that one AP could only access to the cloud server, the collocated edge server if existed, and neighbor edge servers (i.e., the pair of edge servers connected via the routing path as illustrated in Fig.1) due to transmission latency limitation. We refer to  $\mathcal{M}_k \subseteq \mathcal{M}$  as the *candidate server set* of the  $k$ -th AP,  $\mathcal{K}_m \subseteq \mathcal{K}$  as the *potential AP set* of the  $m$ -th edge server, and  $\rho_{k,m}$  as the collocation indicator (i.e.,  $\rho_{k,m} = 1$  if  $k$ -th AP and  $m$ -th edge server collocated, otherwise  $\rho_{k,m} = 0$ ) ( $\forall k \in \mathcal{K}, m \in \mathcal{M}$ ). We consider cloud server as one special processing server with stronger computation capability, but also with larger transmission latency compared with the edge servers.

The dispatchers are implemented on the APs in a distributed manner. Without loss of generality, it is assumed that there are  $J$  types of jobs computed in this system, which are denoted via the set  $\mathcal{J} \triangleq \{1, \dots, J\}$ . The time axis is organized by time slots. The arrivals of the type- $j$  jobs at the  $k$ -th AP in different time slots are assumed to be independent and identically distributed Bernoulli random variables, and the arrival probability are denoted as  $\lambda_{k,j}$ . Each AP immediately dispatches each type of arrived jobs to one processing server. It's assumed that the network traffic over the routing path is random, and the job uploading from one AP to one edge server consumes a random number of time slots. Let  $\mathbb{U}_{k,m,j}(\Xi)$  be the uploading latency distribution of the type- $j$  jobs from the  $k$ -th AP to the  $m$ -th server with finite support  $\{1, \dots, \Xi\}$ , whose expectation is denoted as  $u_{k,m,j}$ . Specifically, the uploading latency is fixed as 0 if  $\rho_{k,m} = 1$  for job uploading to the collocated edge server ( $\forall k \in \mathcal{K}, m \in \mathcal{M}, j \in \mathcal{J}$ ).

We adopt the *unrelated machines assumption* as in [10] for job computation process, where the computation time on different processing servers would follow independent distribution. Specifically, there are  $J$  parallel virtual machines (VMs) running on each processing server for the  $J$  job types, respectively. It is assumed that the computation time of different job types on different edge servers follows independent memoryless geometric distribution with different expectations as in [16]. Let  $\mathbb{G}(1/c_{m,j})$  be the distribution of the computation time slots for the type- $j$  jobs on the  $m$ -th processing server, where  $\mathbb{G}$  denotes the geometric distribution,  $c_{m,j}$  is the expectation, and  $1/c_{m,j}$  represents the parameter of geometric distribution. For each job type, the uploaded jobs are computed in a First-Come-First-Serve (FCFS) manner, and a processing queue with a maximum job number  $L_{max}$  is established for each VM. The arrival jobs will be discarded on processing server when the processing queue is full.

### B. Signaling Mechanism with Periodic Broadcast

In order to facilitate distributed dispatching for the APs, the signaling mechanism with periodic broadcast is introduced. We refer to every  $t_B$  time slots as a broadcast interval. At the beginning of each broadcast interval, the local state information (LSI) of APs and processing servers are broadcast, and each AP updates its dispatching strategy of job dispatching

when observing the broadcast LSIs from some APs and processing servers. The contents of the LSI at the APs and processing servers are given in the following definitions.

**Definition 1** (LSI of APs). Let  $R_{m,j}^{(k)}(\xi, t, n) \in \{0, 1\}$  be the indicator of the type- $j$  jobs at the  $n$ -th time slot of the  $t$ -th interval. Its value is 1 when there is one job being uploaded from the  $k$ -th AP to the  $m$ -th server which has been delivered for  $\xi$  time slots, and 0 otherwise ( $\forall k \in \mathcal{K}, m \in \mathcal{M}, j \in \mathcal{J}$ ). Let  $\omega_{k,j}(t)$  be the target processing server for the type- $j$  jobs of the  $k$ -th AP dispatched at the beginning of the  $t$ -th broadcast interval. And the LSI of the  $k$ -th AP at the beginning of the  $t$ -th broadcast interval is defined as

$$\mathcal{R}_k(t) \triangleq \left( \left\{ R_{m,j}^{(k)}(t, 0) \mid \forall m \in \mathcal{M}, j \in \mathcal{J} \right\}, \mathcal{A}_k(t) \right), \quad (1)$$

where

$$\mathbf{R}_{m,j}^{(k)}(t, 0) \triangleq \left( R_{m,j}^{(k)}(0, t, 0), \dots, R_{m,j}^{(k)}(\Xi, t, 0) \right), \quad (2)$$

$$\mathcal{A}_k(t) \triangleq \left\{ \omega_{k,j}(t) \mid \forall j \in \mathcal{J} \right\} \quad (3)$$

are referred as *status of the type- $j$  job from the  $k$ -th AP to the  $m$ -th processing server*, and *dispatching actions of the  $k$ -th AP at the beginning of the  $t$ -th broadcast interval*, respectively.

**Definition 2** (LSI of Processing Servers). Let  $Q_{m,j}(t, n)$  be the number of type- $j$  jobs on the  $m$ -th processing server at the  $n$ -th time slot of the  $t$ -th interval ( $\forall m \in \mathcal{M}, j \in \mathcal{J}$ ). The LSI of the  $m$ -th processing server at beginning of the  $t$ -th broadcast interval is defined as

$$\mathcal{Q}_m(t) \triangleq \left\{ Q_{m,j}(t, 0) \mid \forall j \in \mathcal{J} \right\}. \quad (4)$$

Moreover, we refer to *global state information* (GSI) as the aggregation of LSIs of all the APs and processing servers in one broadcast interval.

**Definition 3** (Global State Information). At the  $t$ -th broadcast interval, *global state information* (GSI) is defined as follows.

$$\mathbf{S}(t) \triangleq \left( \left\{ \mathcal{R}_k(t) \mid \forall k \in \mathcal{K} \right\}, \left\{ \mathcal{Q}_m(t) \mid \forall m \in \mathcal{M} \right\} \right). \quad (5)$$

As a remark notice that the observable LSI may be outdated due to signaling latency among APs and processing servers. As the APs and processing servers may reside in different locations of a MAN, the transmission latency of LSI is not negligible. It might be inefficient for one AP (say the  $k$ -th AP) to collect the complete GSI before updating the dispatching policy. For example, the transmission latency of the LSI from the edge servers out of its *candidate server set*  $\mathcal{M}_k$  may be large, and some broadcast information may be discarded by the routers after a certain number of hops. Here, we firstly define *conflict AP set* and notice that only the LSIs of a subset of APs are interested for one single dispatcher, and then we define the interested partially-observable information as *observable state information* (OSI) based on the *conflict AP set* and *candidate server set*. The definitions of *conflict AP set* and OSI are given below, respectively.

**Definition 4** (Conflict AP Set). The *conflict AP set* to the  $k$ -th AP consists of the neighboring APs who have direct impacts on

the queueing time of the jobs on processing servers dispatched from the  $k$ -th AP, i.e.,  $\mathcal{X}_k \triangleq \bigcup_{m \in \mathcal{M}_k} \mathcal{K}_m$ .

**Definition 5** (Observable State Information). *The observable state information (OSI) of the  $k$ -th AP ( $\forall k \in \mathcal{K}$ ) at the  $t$ -th broadcast interval is defined as the aggregation of LSIs of the APs in conflict AP set and the edge servers in candidate server set of the  $k$ -th AP, i.e.,*

$$\mathbf{S}_k(t) \triangleq \left( \left\{ \mathcal{R}_{k'}(t) \mid \forall k' \in \mathcal{X}_k \right\}, \left\{ \mathcal{Q}_m(t) \mid \forall m \in \mathcal{M}_k \right\} \right). \quad (6)$$

The  $k$ -th AP is able to collect its OSI  $\mathbf{S}_k(t)$  at the  $\mathcal{D}_k(t)$ -th time slots of the  $t$ -th broadcast interval, where  $\mathcal{D}_k(t)$  denotes the signaling latency of the  $k$ -th AP at the  $t$ -th broadcast interval which is a random variable in the unit of timeslot. It is assumed that  $\mathcal{D}_k(t)$  follows identical and independent distribution in different broadcast interval.

#### IV. POMDP-BASED PROBLEM FORMULATION

In this section, we formulate the optimization of job dispatching at all APs as a Markov decision process (MDP) problem. Since each AP updates the job dispatching actions according to OSI instead of GSI, the MDP problem is a partially observable MDP (POMDP). Firstly, we give the definitions of *dispatching policy* and *cost function*, together with the *system state* (i.e., the GSI) defined previously, to complete the MDP problem formulation.

**Definition 6** (Dispatching Policy). *The individual dispatching policy of the  $k$ -th AP, denoted as  $\Omega_k$  ( $\forall k \in \mathcal{K}$ ), maps from its OSI  $\mathbf{S}_k$  and its signaling latency  $\mathcal{D}_k$  to the dispatching action for each job type, i.e.,*

$$\Omega_k(\mathbf{S}_k(t), \mathcal{D}_k(t)) \triangleq \mathcal{A}_k(t+1) \quad (7)$$

The aggregation of individual policy of all APs is referred to as the system dispatching policy  $\Omega$ . Thus,

$$\Omega(\mathbf{S}(t), \mathcal{D}(t)) \triangleq \left\{ \Omega_1(\mathbf{S}_1(t), \mathcal{D}_1(t)), \dots, \Omega_K(\mathbf{S}_K(t), \mathcal{D}_K(t)) \right\}, \quad (8)$$

where  $\mathcal{D}(t) \triangleq \{\mathcal{D}_1(t), \dots, \mathcal{D}_K(t)\}$ .

According to the Little's law [17], the average response time per job, counting the number of broadcast intervals from job arrival to the accomplishment of computation, is proportional to the number of jobs in the system, given the job arrival rates at all the APs. Hence, we define the cost function per broadcast interval, which consists the cumulative cost to be minimized in the MDP problem, as follows.

**Definition 7** (Cost Function per Broadcast Interval). *The cost function of the  $t$ -th broadcast interval with  $\mathbf{S}(t)$  is defined as*

$$g(\mathbf{S}(t)) \triangleq \sum_{m \in \mathcal{M}, j \in \mathcal{J}} \left\{ \sum_{k \in \mathcal{K}} \|\mathbf{R}_{m,j}^{(k)}(t, 0)\|_1 + Q_{m,j}(t, 0) + \beta \cdot I[Q_{m,j}(t, 0) = L_{max}] \right\}, \quad (9)$$

where  $\|\mathbf{x}\|_1$  denotes the  $L^1$ -norm of the vector  $\mathbf{x}$ ,  $\beta$  is the weight of overflow penalty, and  $I[\cdot]$  denotes the indicator

function which is equal to 1 when the inner statement is true and 0 otherwise.

Then, since the job dispatching in one broadcast interval will affect the cost of the successive broadcast intervals, we should consider the joint minimization of the cumulative costs of all the broadcast intervals. Specifically, we consider the following discounted sum of the costs of all the broadcast intervals as the system objective.

$$\bar{G}(\mathbf{S}, \Omega) \triangleq \lim_{T \rightarrow \infty} \mathbb{E}_{\{\mathbf{S}(t) \mid \forall t\}}^{\Omega} \left[ \sum_{t=1}^T \gamma^{t-1} g(\mathbf{S}(t)) \mid \mathbf{S}(1) \right], \quad (10)$$

where  $\mathbb{E}_{\{\mathbf{S}(t) \mid \forall t\}}^{\Omega}[\cdot]$  denotes the expectation with respect to all possible system states in the future given scheduling policy  $\Omega$ , and  $\gamma \in (0, 1)$  is the discount factor. Hence, the optimization of job dispatching policy can be formulated as the following minimization problem.

$$\mathbf{P1}: \min_{\Omega} \bar{G}(\mathbf{S}, \Omega). \quad (11)$$

Finally, if the GSI  $\mathbf{S}(t)$  and signaling latency  $\mathcal{D}(t)$  are known to all the APs, the MDP in problem P1 can be solved via the following Bellman's equations as in [18]:

$$V(\mathbf{S}(t)) = g(\mathbf{S}(t)) + \gamma \mathbb{E}_{\mathcal{D}} \left\{ \min_{\Omega(\mathbf{S}(t), \mathcal{D}(t))} \sum_{\mathbf{S}(t+1)} \Pr \left\{ \mathbf{S}(t+1) \mid \mathbf{S}(t), \Omega(\mathbf{S}(t), \mathcal{D}(t)) \right\} \cdot V(\mathbf{S}(t+1)) \right\}, \quad (12)$$

where the value function  $V(\mathbf{S}(t))$  of the optimal policy  $\Omega^*$  is defined as follows.

$$V(\mathbf{S}(t)) \triangleq \lim_{T \rightarrow \infty} \mathbb{E}_{\{\mathbf{S}(t) \mid \forall t\}}^{\Omega^*} \left[ \sum_{t=1}^T \gamma^{t-1} g(\mathbf{S}(t)) \mid \mathbf{S}(1) \right]. \quad (13)$$

Moreover, the optimal policy  $\Omega^*$  can be obtained by solving the right-hand-side (RHS) of the above Bellman's equations.

However, it is infeasible to solve the above Bellman's equations because each AP only has the knowledge of its own OSI and local signaling latency in our considered edge computing system. Thus problem P1 is actually a POMDP, whose general solution is of huge complexity [19], [20].

#### V. DISTRIBUTED ALGORITHM WITH PARTIAL INFORMATION

In this section, we shall introduce a novel approximation method to decouple the centralized optimization on the RHS of the Bellman's equations in equation (12) to each AP for arbitrary system state. For the APs outside the conflict AP sets of each other, the update of dispatching actions at one AP will not affect the task computation originated from other APs. On the other hand, for the APs within the same conflict AP set, the optimization of their dispatching actions is coupled. Hence, we introduce an alternative actions update algorithm to optimize the dispatching actions of subset of  $\mathcal{K}$  in each broadcast interval periodically, while other APs maintain their dispatching actions in the previous broadcast interval. Specifically, the proposed distributed algorithm consists of the following two steps:

- 1) We first introduce a time-variant baseline policy, use its value function to approximate the value function of

the optimal policy  $\Omega^*$  in each broadcast interval, and derive the analytical expression of the approximate value function for arbitrary GSI in Section V-A.

- 2) With the approximate value function, an alternative actions update algorithm, where only a subset of APs are selected to update their dispatching actions distributedly in each broadcast interval, is proposed in Section V-B. Moreover, the analytical performance bound is derived in Section V-C.

#### A. Baseline Policy and Approximate Value Function

To alleviate the curse of dimensionality, we first use the baseline policy with fixed dispatching actions to approximate the value function at the RHS of the Bellman's equations in equation (13). The baseline policy is elaborated below.

**Policy 1** (Baseline Policy). In the baseline policy  $\Pi$ , each AP fixes its dispatching actions as in the previous broadcast interval. Specifically, at the  $t$ -th broadcast interval,

$$\Pi(\mathbf{S}(t), \mathcal{D}(t)) \triangleq \left\{ \Pi_1(\mathbf{S}_1(t), \mathcal{D}_1(t)), \dots, \Pi_K(\mathbf{S}_K(t), \mathcal{D}_K(t)) \right\}, \quad (14)$$

where

$$\Pi_k(\mathbf{S}_k(t), \mathcal{D}_k(t)) \triangleq \left\{ \omega_{k,j}(t+1) \mid \forall j \in \mathcal{J} \right\}, \forall k \in \mathcal{K}. \quad (15)$$

Let  $W_\Pi(\cdot)$  be the value function of the baseline policy  $\Pi$ , we shall approximate the value function of the optimal policy  $V(\cdot)$  via  $W_\Pi$ , i.e.,

$$\begin{aligned} V(\mathbf{S}(t+1)) &\approx W_\Pi(\mathbf{S}(t+1)) \\ &= \sum_{m \in \mathcal{M}, j \in \mathcal{J}} \left\{ \sum_{k \in \mathcal{K}} \tilde{W}_{k,m,j}^\dagger(\mathbf{S}(t+1)) + \tilde{W}_{m,j}^\ddagger(\mathbf{S}(t+1)) \right\}, \end{aligned} \quad (16)$$

where  $\tilde{W}_{k,m,j}^\dagger(\mathbf{S}(t+1))$  denotes the cost raised by the type- $j$  jobs which are being transmitted from the  $k$ -th AP to the  $m$ -th processing server with the baseline policy  $\Pi$  and initial system state  $\mathbf{S}(t+1)$ , and  $\tilde{W}_{m,j}^\ddagger(\mathbf{S}(t+1))$  denotes the cost raised by the type- $j$  jobs on the  $m$ -th server. Their definitions are given below.

$$\tilde{W}_{k,m,j}^\dagger(\mathbf{S}(t+1)) \triangleq \sum_{i=0}^{\infty} \gamma^{i+1} \mathbb{E}^\Pi \left[ \|\mathbf{R}_{m,j}^{(k)}(t+i+1)\|_1 \right], \quad (17)$$

$$\begin{aligned} \tilde{W}_{m,j}^\ddagger(\mathbf{S}(t+1)) &\triangleq \sum_{i=0}^{\infty} \gamma^{i+1} \mathbb{E}^\Pi \left[ Q_{m,j}(t+i+1) + \right. \\ &\quad \left. \beta I[Q_{m,j}(t+i+1) = L_{max}] \right]. \end{aligned} \quad (18)$$

#### B. Distributed Actions Update

Although the optimal value function has been approximated via the baseline policy in the previous part, it is still infeasible to solve the RHS of the Bellman's equations as the evaluation of equation (17) and (18) requires the knowledge of GSI and signaling latency at all APs. Instead, it is feasible for part of APs to update their dispatching actions distributedly in each broadcast interval and achieve a better performance compared with baseline policy. Hence, we first define the following

sequence of AP subsets, where each subset are selected to update dispatching actions periodically.

**Definition 8** (Subset Partition). Let  $\mathcal{Y}_1, \dots, \mathcal{Y}_N \subseteq \mathcal{K}$  be a collection of subsets of AP set  $\mathcal{K}$ , which satisfy

$$\bigcup_{n=0, \dots, N-1} \mathcal{Y}_n = \mathcal{K} \quad (19)$$

$$\mathcal{M}_y \cap \mathcal{M}_{y'} = \emptyset, y' \neq y \ (\forall y', y \in \mathcal{Y}_n). \quad (20)$$

The subset partition is not trivial and a partition to minimize the update period  $N$  is preferred. A heuristic greedy algorithm is given in Algorithm 1.

---

#### Algorithm 1: Greedy Subset Partition Algorithm

---

**Input:**  $\mathcal{K}, \{\mathcal{M}_k, \forall k \in \mathcal{K}\}$

**Output:** a subset partition  $\{\mathcal{Y}_n\}$

- 1 Initialize a subset partition as  $\mathcal{Y}_n = \{n\} \ (\forall n \in \mathcal{K})$ .
  - 2 **while**  $\exists \mathcal{Y}_a$  and  $\mathcal{Y}_b \ (a \neq b)$  that  $\cup\{\mathcal{M}_y | y \in \mathcal{Y}_a\} \cap \cup\{\mathcal{M}_y | y \in \mathcal{Y}_b\} \neq \emptyset$  **do**
  - 3     Count number of subsets in the current subset partition which have disjoint candidate set with  $\mathcal{Y}_n \ (\forall n)$ , denoted the number as  $I_n$ .
  - 4      $\tilde{n} \leftarrow \arg \min_n I_n$
  - 5     Merge subset  $\mathcal{Y}_{\tilde{n}}$  with one of its disjoint subsets.
- 

At the  $t$ -th broadcast interval, the APs in the subset indexed with  $n \triangleq t \pmod{N}$  update their dispatching actions, while the other APs keep the same dispatching actions as the previous broadcast interval. Hence, let

$$\tilde{\mathcal{A}}(t) \triangleq \left\{ \omega_{y,j}(t+1) \mid \forall y \in \mathcal{Y}_n, j \in \mathcal{J} \right\} \quad (21)$$

be the aggregation of dispatching actions of the APs in the subset  $\mathcal{Y}_n$ , and

$$\hat{\mathcal{A}}(t) \triangleq \left\{ \omega_{y,j}(t+1) \mid \forall y \notin \mathcal{Y}_n, j \in \mathcal{J} \right\} \quad (22)$$

be the aggregation of dispatching actions of the rest APs, which are same as in the previous broadcast interval. At the  $t$ -th broadcast interval, the optimization of dispatching actions  $\hat{\mathcal{A}}(t)$  at the RHS of the Bellman's equations can be rewritten as the following problem.

$$\mathbf{P2:} \min_{\hat{\mathcal{A}}(t)} \sum_{\mathbf{S}(t+1)} \Pr \left\{ \mathbf{S}(t+1) \mid \mathbf{S}(t), \hat{\mathcal{A}}(t), \tilde{\mathcal{A}}(t) \right\} \cdot W_\Pi(\mathbf{S}(t+1)). \quad (23)$$

Moreover, we have the following conclusion on the decomposition of P2.

**Lemma 1.** The optimization problem in P2 can be equivalently decoupled into local optimization problems at APs for each subset partition. Specifically, the local optimization for the  $y$ -th AP in the  $n$ -th subset ( $\forall n$ ) can be written as

$$\begin{aligned} \mathbf{P3:} \min_{\mathcal{A}_y(t+1)} &\mathbb{E}_{\{\mathbf{S}_y(t+1) \mid \mathbf{S}_y(t), \hat{\mathcal{A}}(t), \mathcal{A}_y(t+1)\}} \\ &\sum_{j \in \mathcal{J}, m \in \mathcal{M}_y} \left\{ \tilde{W}_{k,j}^\dagger(\mathbf{S}_y(t+1)) + \tilde{W}_{m,j}^\ddagger(\mathbf{S}_y(t+1)) \right\}. \end{aligned} \quad (24)$$

*Proof.* At the  $t$ -th broadcast interval, the  $y$ -th AP in the subset  $\mathcal{Y}_n$  updates its dispatching actions, which could only affect the future cost raised on itself and its corresponding *candidate server set*, i.e., the part of its OSI. Hence, it's obvious that the equation (17) and equation (18) on the RHS of the Bellman's equations could be reduced into the form based only on the OSI of the  $y$ -th AP ( $\forall y \in \mathcal{Y}_n$ ).  $\square$

The optimization of dispatching actions  $\mathcal{A}_y(t+1)$  for the  $y$ -th AP ( $\forall y \in \mathcal{Y}_n$ ) in P3 could be achieved via searching all the processing servers in  $\mathcal{M}_y$ , whose computational complexity is  $O(J|\mathcal{M}_y|)$ . As a result, the overall algorithm of job dispatching is elaborated in Algorithm 2.

---

**Algorithm 2:** Online Alternative Actions Update Algorithm

---

- 1 Initialize all the APs with heuristic dispatching actions  $\{\omega_{k,j}(0) | \forall k \in \mathcal{K}, j \in \mathcal{J}\}$ .
  - 2 **for**  $t = 0, 1, 2, \dots$  **do**
  - 3    $n \leftarrow t \pmod{N}$
  - 4   **for**  $y \in \mathcal{Y}_n$  **do in parallel**
  - 5     The  $y$ -th AP observes  $\mathbf{S}_y(t)$  after  $\mathcal{D}_y(t)$ .
  - 6     Solve P3 with  $\mathbf{S}_y(t), \mathcal{D}_y(t)$  and obtain optimized actions  $\{\tilde{\omega}_{y,j}(t+1) | \forall j \in \mathcal{J}\}$
  - 7    $\tilde{\mathcal{A}}(t+1) \leftarrow \{\tilde{\omega}_{y,j}(t+1) | \forall y \in \mathcal{Y}_n, j \in \mathcal{J}\}$ .
  - 8    $\hat{\mathcal{A}}(t+1) \leftarrow \{\omega_{y,j}(t) | \forall y \in \mathcal{Y}_{n-1}, j \in \mathcal{J}\} \cup \{\omega_{y,j}(t) | \forall y \notin \mathcal{Y}_{n-1}, j \in \mathcal{J}\}$
  - 9
- 

As a remark notice that since in Algorithm 2, the computation complexity at each AP scales linearly with respect to the size of candidate edge server set, it can be deployed in a scenario with massive APs and edge servers, as long as the available number of edge servers for each AP is limited.

### C. Analytical Performance Bound

In most of the existing approximate MDP solutions [21]–[26], the performance is difficult to bound analytically as the approximate value function has no accurate meaning on the system cost or utility. In the proposed algorithm, however, we derive the analytical expression for the baseline policy as the approximate. Hence, the alternative dispatching actions update can ensure to achieve a better performance than the baseline policy. This conclusion is summarized below.

**Lemma 2** (Analytical Cost Upper Bound). *Let  $W_{\tilde{\Omega}}(\cdot)$  be the value function (average cost) of the proposed policy  $\tilde{\Omega}$ , i.e.,*

$$W_{\tilde{\Omega}}(\mathbf{S}) \triangleq \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}_{\{\mathbf{S}(t) | \forall t\}}^{\tilde{\Omega}} \left[ g(\mathbf{S}(t)) | \mathbf{S}(1) = \mathbf{S} \right], \quad (25)$$

*we have*

$$V(\mathbf{S}) \leq W_{\tilde{\Omega}}(\mathbf{S}) \leq W_{\Pi}(\mathbf{S}), \forall \mathbf{S}. \quad (26)$$

*Proof.*  $V(\mathbf{S}) \leq W_{\tilde{\Omega}}(\mathbf{S})$  is straightforward as  $\tilde{\Omega}$  is not optimal policy. The proof of  $W_{\tilde{\Omega}}(\mathbf{S}) \leq W_{\Pi}(\mathbf{S})$  is equivalent to prove the improvement of one-step policy iteration, which is similar to the proof of *Policy Improvement Property* in [27].  $\square$

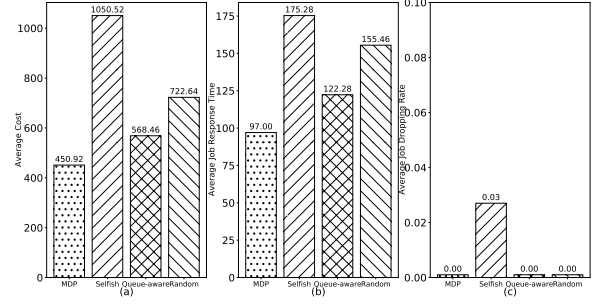


Fig. 2. Illustration of performance metrics comparison with benchmarks.

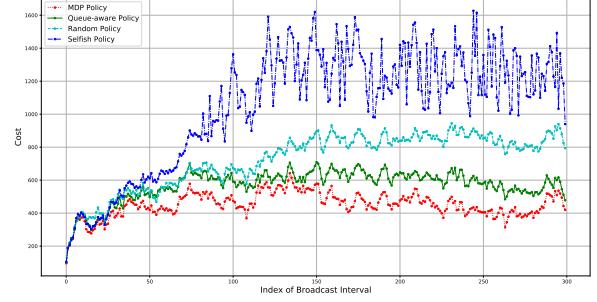


Fig. 3. Illustration of cost versus index of broadcast interval.

Therefore,  $W_{\Pi}(\mathbf{S})$  derived in equation (16) can be used as the analytical cost upper bound of the proposed policy  $\Pi$ . Moreover, Lemma 2 also implies that the proposed policy with fixed service edge server for each AP, as long as the static dispatching policy is used as the baseline policy.

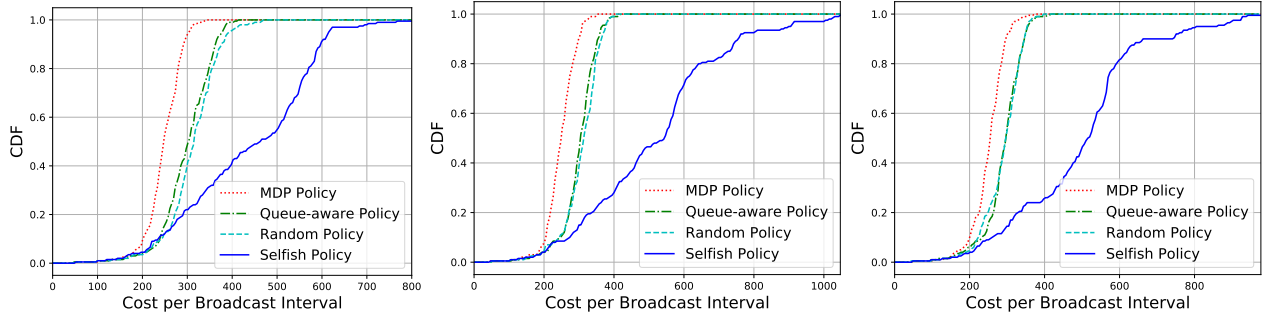
## VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed low-complexity dispatching policy  $\tilde{\Omega}$  by numerical simulations. The experiment setup and performance benchmarks are elaborated in Section VI-A. The simulation results are illustrated in Section VI-B. The sensitivity study on parameters is also applied to provide some insights on the robustness of the proposed policy in Section VI-C.

### A. Experiment Setup

In the simulation, we assume that there are  $K = 15$  APs,  $M = 10$  processing servers and  $J = 10$  types of jobs in the system. One broadcast interval is consist of  $t_B = 25$  time slots. The network topology among APs is generated according to Barabási-Albert (BA) model [28] and the processing servers are randomly placed collocated with the APs. The arrival traces and job processing time for each job type are extracted from Google cluster traces [29] and then randomly assigned on APs and edge servers, respectively. The maximum uploading latency is  $\Xi = 3t_B$ , and the distribution of  $\mathbb{U}_{k,m,j}(\Xi)$  ( $\forall k \in \mathcal{K}, m \in \mathcal{M}_k, j \in \mathcal{J}$ ) is arbitrarily generated within the support  $\{0, 1, \dots, \Xi\}$ . The signaling latency is with an integer support from  $0.7t_B$  to  $0.9t_B$  time slots. Each queue for VMs on edge server is with maximum queue length  $L_{max} = 50$ ,





(a) signaling latency as 5 time slots. (b) signaling latency as 12 time slots. (c) signaling latency as 25 time slots.

Fig. 4. Algorithm Robustness versus Signaling Latency.

i.e., there would be at most 50 jobs on one edge server. The discount factor  $\gamma$  is 0.95 and the overflow penalty  $\beta$  is 120.

We also propose three heuristic benchmarks to profile the performance of the proposed MDP policy.

- **Random Dispatching Policy:** Randomly choose a dispatching edge server in each time slot;
- **Selfish Policy:** Always choose the edge server with the minimum sum of the expected uploading time and processing time;
- **Queue-aware Policy:** Always choose the edge server with the minimum sum of expected uploading time, processing time and queueing time based on the observation of outdated queue states.

Moreover, we choose the Selfish Policy as the initial dispatching actions for our proposed algorithm (Algorithm 2).

### B. Performance Analysis

As illustrated in Fig.2(a), the proposed algorithm (MDP Policy) outperforms all the benchmarks in the average system cost. More insights on the performance comparison are provided in Fig.2(b) and (c). In the former figure, the average job response times, measuring the average number of broadcast intervals from job's arrival at one AP to the completeness of computation at one edge server, are compared. It can be observed that the proposed policy still outperforms all the benchmarks. In Fig.2(c), the job dropping rates, measuring the ratio of jobs dropped by edge servers due to queue overflow, are also compared. It is shown that the proposed policy outperforms other three benchmarks with the minimum average cost and job response time. And there is no dropping jobs incurred compared with the Selfish policy, which is the initial baseline policy for our proposed algorithm. Finally, an realization of job dispatching is illustrated in Fig.3, where the number of jobs in the system is plot versus the index of broadcast interval. It can be observed that the proposed policy manage to keep the number of jobs in lower level, compared with the other benchmarks. This demonstrates its high dispatching efficiency.

### C. Sensitivity Study

**Signaling Latency.** The simulation results with different signaling latency  $D_k$  ( $\forall k \in \mathcal{K}$ ) are illustrated in Fig.4, where

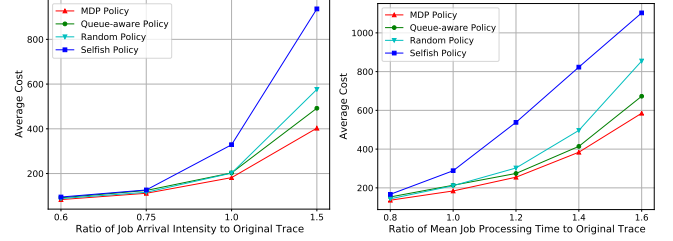


Fig. 5. Illustration of average system cost versus job arrival intensity.

the cumulative distribution function (CDF) of the job number in the system is plotted. Specifically, the signaling latency of all the APs is set to 5, 12, 25 in Fig.4(a), Fig.4(b), Fig.4(c), respectively. It can be observed that with the increasing of signaling latency, the performance of Queue-aware Policy becomes worse. The Queue-aware policy slightly outperforms the Random Policy in Fig.5(a) with smaller signaling latency (achieving a smaller number of jobs in the system), and becomes worse in Fig.5(c) with large signaling latency. This demonstrates that the Queue-aware Policy is sensitive to signaling latency. In all the figures, the proposed policy outperforms all the benchmarks, which demonstrates its robustness versus signaling latency.

**Job Arrival Intensity.** We carry out the sensitivity study of job arrival intensity by integer scaling the interval of jobs arriving in Google cluster traces. The average system cost versus the number of APs is illustrated in Fig.5. With the increasing of job arrival intensity, the average system cost increases in all the benchmarks and our proposed policy. It can be observed that our policy performs the best. Moreover, the performance gain becomes significant when the computation load is heavy.

**Mean Processing Time.** The simulation results of various mean processing time are illustrated in Fig.6, where the mean processing time is taken as  $c_{m,j}$  of the processing time distribution  $\mathbb{G}(1/c_{m,j})$  in our computation model assumption. Generally speaking, with the increasing average processing time, the average system cost increases in all the benchmarks and our proposed policy. The simulation results are consistent with that in Fig.5.

## VII. CONCLUSION

In this paper, we consider an online distributed job dispatcher design problem for an edge computing system residing in a Metropolitan Area Network. In this edge computing system, the job dispatchers are implemented in a distributed manner on multiple access points (APs) which collect jobs from mobile users and then dispatch jobs to one edge server or cloud server for processing. To facilitate the cooperation among distributed job dispatchers, a signaling mechanism is introduced where the APs and edge servers would periodically broadcast their local state information to the job dispatchers. However, the reception of updated and fully-observed global system state is discouraged as the transmission latency is non-negligible in MAN and reception of all broadcast is time consuming. Hence, we formulate the distributed optimization problem of job dispatching strategies as a POMDP problem, with outdated and partially-observable information. The conventional solution for POMDP is impractical due to huge time complexity. In this paper, we propose a novel low-complexity solution framework for distributed job dispatching, based on which the optimization of job dispatching policy can be decoupled via an alternative policy iteration algorithm and a theoretical performance lower bound is obtained. The evaluation results show that our proposed policy can achieve obvious and robust performance gain compared with heuristic baselines. Furthermore, this work assumes available knowledge on the distributions of signaling latency, uploading latency and computation time. As an extension, the reinforcement learning could be integrated with the proposed solution framework when the above statistics are absent.

## REFERENCES

- [1] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, Fourthquarter 2018.
- [2] A. Gember, A. Akella, J. Pang, A. Varshavsky, and R. Caceres, "Obtaining in-context measurements of cellular network performance," in *2012 Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 287–300.
- [3] D. Loghin, L. Ramapantulu, and Y. M. Teo, "Towards analyzing the performance of hybrid edge-cloud processing," in *2019 IEEE International Conference on Edge Computing (EDGE)*, 2019, pp. 87–94.
- [4] Z. Zhou, X. Chen, W. Wu, D. Wu, and J. Zhang, "Predictive online server provisioning for cost-efficient iot data streaming across collaborative edges," in *the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '19)*. New York, NY, USA: ACM, 2019, pp. 321–330.
- [5] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based iot," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, June 2018.
- [6] C. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, June 2019.
- [7] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, March 2019.
- [8] 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 Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, Nov 2017.
- [9] —, "Distributed online optimization of fog computing for selfish devices with out-of-date information," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7704–7717, Nov 2018.
- [10] H. Tan, Z. Han, X. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.
- [11] S. Huang, B. Lv, and R. Wang, "Mdp-based scheduling design for mobile-edge computing systems with random user arrival," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.
- [12] S. Huang, B. Lv, R. Wang, and K. Huang, "Scheduling for mobile edge computing with random user arrivals — an approximate mdp and reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2020.
- [13] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.
- [14] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.
- [15] B. Lyu, Y. Hong, H. Tan, Z. Han, and R. Wang, "Cooperative jobs dispatching in edge computing network with unpredictable uploading delay," *Journal of Communications and Information Networks*, vol. 5, no. 1, pp. 75–85, 2020.
- [16] S. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5225–5240, Aug 2018.
- [17] J. D. C. Little, "A proof for the queueing formula:  $L = \lambda w$ ," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, Jun. 1961.
- [18] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.
- [19] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized pomdps: Towards efficient policy computation for multiagent settings," in *the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 705–711.
- [20] C. Boutilier, "Sequential optimality and coordination in multiagent systems," in *the 16th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 478–485.
- [21] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. M. Lau, "Dynamic virtual machine management via approximate markov decision process," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.
- [22] B. Lv, L. Huang, and R. Wang, "Joint downlink scheduling for file placement and delivery in cache-assisted wireless networks with finite file lifetime," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4177–4192, 2019.
- [23] Z. Han, H. Tan, R. Wang, S. Tang, and F. C. M. Lau, "Online learning based uplink scheduling in hetnets with limited backhaul capacity," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2348–2356.
- [24] Q. Lan, B. Lv, R. Wang, K. Huang, and Y. Gong, "Adaptive video streaming for massive mimo networks via approximate mdp and reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 9, pp. 5716–5731, 2020.
- [25] B. Lv, R. Wang, Y. Cui, Y. Gong, and H. Tan, "Joint optimization of file placement and delivery in cache-assisted wireless networks with limited lifetime and cache space," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2339–2354, 2020.
- [26] S. Huang, B. Lv, R. Wang, and K. Huang, "Scheduling for mobile edge computing with random user arrivals—an approximate mdp and reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7735–7750, 2020.
- [27] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 2000.
- [28] R. Albert, H. Jeong, and A.-L. Barabási, "Diameter of the world-wide web," *nature*, vol. 401, no. 6749, pp. 130–131, 1999.
- [29] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014-11-17 for version 2.1. Posted at <https://github.com/google/cluster-data>.