Quality-Aware Video Offloading in Mobile Edge Computing: A Data-driven Two-stage Stochastic Optimization

Weibin Ma, Lena Mashayekhy
Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716, USA
{weibinma, mlena}@udel.edu

***Abstract*—Most camera-based mobile devices require ultra low-latency video analytics such as object detection and action recognition. These devices face severe resource constraints, and thus, video offloading to Mobile Edge Computing (MEC) seems a reasonable solution. However, MEC is facing several key challenges–especially due to uncertainties caused by dynamic device mobility–to provide efficient video offloading solutions that enable both maximum performance for video analytics and minimum latency. In this paper, we study the Video Offloading Problem (VOP) in MEC in detail to address these challenges. We formulate VOP as a Two-stage Stochastic Program, called TSP-VOP, to model the uncertainties in the environment. We propose a novel clustering-based Sample Average Approximation to effectively solve TSP-VOP in uncertain dynamic environments, while satisfying the required latency. We perform extensive experiments to validate the effectiveness of our proposed algorithm.**

***Index Terms*—Mobile Edge Computing, Video Offloading, Mobility, Video Quality, Two-Stage Stochastic Program, Clustering.**

I. Introduction

It is predicted that videos will account for 79% of the world's mobile data traffic by 2022 [1]. Many camera-based mobile devices (e.g., surveillance drones and vehicle dash cameras) require real-time analytics of high-resolution video streams such as object detection and action recognition. Mobile Edge Computing (MEC) has recently been introduced as an emerging solution that enables mobile devices to offload their delay-sensitive tasks such as video analytics to physically proximal mini-datacenters, called cloudlets, in order to improve the quality of service (QoS) [2].

Differing from conventional computation offloading [3], video offloading in MEC brings its own unique challenges. The performance of video analytics is greatly affected by the quality of videos [4], defined in terms of the number of frames captured per second and the size of frames. To maximize the performance of video analytics (e.g., increase object detection accuracy, detect more objects), a higher video quality is beneficial [5]. However, to reduce latency and satisfy QoS while offloading, a lower video quality should be selected. Therefore, there is a trade-off between video quality and latency that needs to be considered in the design of video offloading mechanisms. A few studies investigated the analytics performance degradation caused by the video quality in offloading [4]–[6]. However, they do not consider the device mobility and service migration possibility in video offloading, since the bandwidth each device received greatly depends on its physical location which could change over time.

Mobility of devices makes video offloading a more challenging problem, especially when the movements of the devices are unknown by the MEC system. When a device moves, keeping its connected cloudlet invariant may greatly increase the communication delay due to its expanded network distance. On the other hand, excessively changing the connected cloudlet could lead to significant migration overhead and massive data movement over MEC. To balance the service performance and migration cost, Sun *et al.* [7] developed a user-centric energy-aware mobility management scheme to optimize the sum of computation and communication delays under the long-term energy budget of the device. Ouyang *et al.* [8] investigated service migration in MEC and proposed a mobility-aware dynamic service placement technique based on Markov approximation. Gao *et al.* [9] proposed an iterative algorithm for offloading with minimum latency considering both cloudlet selection and access point selection. However, these approaches are not suitable for video offloading, since they do not consider video quality to improve the performance of video analytics. Moreover, these approaches are designed for the long-term cost minimization, whereas the performance of video quality only depends on the occurrence of service migration between two consecutive time slots.

In this paper, we propose and formulate the Video Offloading Problem (VOP) as a Two-stage Stochastic Program with recourse, called TSP-VOP, due to uncertainties caused by the dynamic movements of mobile devices. Stochastic programming approach is a promising solution for modeling optimization problem that involves uncertainty [10]. The goal of TSP-VOP is to find optimal offloading decisions for all mobile devices to offload their videos with maximum quality and minimum migration cost, while other desirable constraints (e.g., latency and energy requirements) are satisfied. However, the formulated TSP-VOP model requires a large number of realizations, called scenarios, to obtain a good representation of the uncertainties for a more accurate estimation. To resolve this issue, we propose a novel Clustering-based Sample Average Approximation Video Offloading Algorithm, called CSAA-VOA, to approximate the expected cost of TSP-VOP with much fewer scenarios. Our proposed CSAA-VOA guarantees computational tractability by reducing the sample size, while it does not negatively impact the quality of the obtained solutions. To the best of our knowledge, this is *the first work that addresses the VOP in MEC by considering uncertain device mobility via exploiting stochastic programming.*

II. System Model and Problem Formulation

A. System Model

We denote a set of cloudlets by $\mathcal{M} = \{1, 2, \ldots, M\}$ and a set of mobile devices capturing videos by $\mathcal{N} = \{1, 2, \ldots, N\}$. To characterize the movements of devices, we consider a time-slotted format, where the time horizon is discretized into multiple time slots $\mathcal{T} = \{1, 2, \ldots, T\}$.

The quality of videos is defined as a function of a *video coding ratio*, defined below. The captured videos are divided into multiple video chunks for offloading [11]. Each chunk is compressed by a specific video coding ratio that is defined as the ratio of the size of a compressed video chunk to the size of the original (uncompressed) video chunk. We denote $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ to represent a set of available video coding ratios to be selected for offloading.

At any time slot $t \in \mathcal{T}$, device $n \in \mathcal{N}$ has a video chunk with a data size τ_n and deadline $\mathbf{d}_n$ to offload. The location of device n at time slot t is specified by its two-dimensional coordinate $(l_n^x(t), l_n^y(t))$. The locations of devices at the current time slot are known by the MEC system, while the locations of devices at the future time slots are unknown by the MEC system. We use $r_{nik}(t)$ to express the selected coding ratio of device n when offloading its video chunk to cloudlet i with coding ratio $a_k \in \mathcal{A}$ at time slot t.

Each cloudlet $i \in \mathcal{M}$ has computing capabilities $\mathbf{C}_i$, bandwidth $\mathbf{B}_i$, and energy consumption capacity $\mathbf{E}_i$. The location of each cloudlet i is fixed and specified by a two-dimensional coordinate (l_i^x, l_i^y). We capture the distance between a device and a cloudlet by using the Euclidean distance. For cloudlet i and device n at time slot t, the distance between them is calculated by $f_{ni}(t) = \sqrt{(l_n^x(t) - l_i^x)^2 + (l_n^y(t) - l_i^y)^2}$.

Video offloading to a cloudlet causes some data transmission delay, called offloading delay. The offloading delay for offloading the video chunk of device n to cloudlet i with a specific coding ratio a_k at time slot t is calculated by:

$$\Lambda_{nik}^r(t) = \frac{\tau_n r_{nik}(t)}{R_{ni}(t)}, \tag{1}$$

where $R_{ni}(t) \triangleq \mathbf{B}_i \log(1 + \frac{p_0\, g_{ni}(t)}{N_0})$ is the transmission rate from device n to cloudlet i at time slot t. Moreover, p_0 is the transmission power, N_0 is the noise power, and $g_{ni}(t)$ is the channel gain at time slot t, which is a function of $f_{ni}(t)$.

We consider energy consumption as a resource limitation for cloudlets. The energy required for executing a video chunk of device n on cloudlet i at time slot t can be expressed as:

$$E_{nik}(t) = \kappa\, \tau_n\, r_{nik}(t) \cdot (\epsilon\, \mathbf{C}_i^2), \tag{2}$$

where κ is the number of required CPU cycles for processing 1 bit of the task and ϵ is the effective switched capacitance of the cloudlet processor.

To avoid excessive service migration, similar to [8], [9], we introduce a *migration cost* into our model. We denote m_{ij} as the migration cost from cloudlet i to cloudlet j, when a service is migrated accordingly.

B. Two-Stage Stochastic Video Offloading Formulation

We formulate VOP as a Two-stage Stochastic Program with recourse, called TSP-VOP, since VOP is not deterministic. TSP-VOP finds the offloading schedules and qualities of the video chunks by maximizing video coding ratios and minimizing their expected migration cost, while explicitly considering future device mobility.

We define the decision variables $X(t)$, $Y(t)$, and $Z(t)$ as the cloudlet allocation decision vector, the migration decision vector, and the video coding ratio decision vector for all devices at time slot t, respectively. More specifically, $x_{ni}(t)$ is 1 if if cloudlet i is allocated to device n for offloading at time slot t, and 0, otherwise; $y_{nij}(t)$ is 1 if the service of device n is migrated from cloudlet i to cloudlet j at time slot t, and 0, otherwise; $z_{nik}(t)$ is 1 if device n offloads video chunk to cloudlet i with coding ratio a_k at time slot t, and 0, otherwise.

In our proposed TSP-VOP model, the offloading decision (assigned cloudlet decision variables $X(t)$ and video coding ratio decision variables $Z(t)$) at current time slot t are defined as the *first stage variables* which have to be decided prior to the realization of a scenario ω (i.e., new locations of devices at time slot $t+1$), that instead is known when the recourse decisions $X(t+1)$, $Y(t+1)$, and $Z(t+1)$ at the second stage are made. A scenario is defined as a possible realization of (uncertain and unknown) future devices' mobility. We assume each scenario ω occurs with probability p_ω.

The objective of the formulated TSP-VOP is to determine the first stage variables that maximizes the video coding ratios and minimizes the migration cost. In other words, the objective is to minimize the sum of negative video coding ratios at the current time slot and mathematical expectation of the recourse cost, which will be defined in the second stage. This objective function is defined as:

TSP-VOP-Stage 1 Objective:

$$\Gamma = \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{M}} \sum_{k\in\mathcal{A}} -a_k\, z_{nik}(t) + \mathbb{E}_p[W^{t+1}(X(t), Z(t), \xi(\omega))] \tag{3}$$

where $\mathbb{E}_p[\cdot]$ is the expected recourse cost under all scenarios generated according to a probability distribution p. In addition, $W^{t+1}(X(t), Z(t), \xi(\omega))$ represents the optimal value of the second stage problem knowing $X(t)$ and $Z(t)$, where $\xi(\omega) = (< L^x(\omega), L^y(\omega) >)$ such that $< L^x(\omega), L^y(\omega) >$ denotes the vector of new locations of devices in a realized scenario ω.

TSP-VOP-Stage 2 Objective: Given the first stage variables $X(t)$ and $Z(t)$ and a realized scenario ω for next time slot $t+1$, we define the second stage objective as follows:

$$W^{t+1}(X(t), Z(t), \xi(\omega)) = \min \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{M}} \sum_{k\in\mathcal{A}} \Big(- a_k\, z_{nik}(t+1) + \beta \sum_{j\in\mathcal{M}} y_{nij}(t+1) m_{ij} \Big) \tag{4}$$

This objective function, or the recourse cost, minimizes the sum of negation of video coding ratios and the migration cost

at time slot $t+1$. In addition, to solve our multi-objective optimization problem and find solutions with the best tradeoff between the conflicting objectives, we introduce a constant coefficient or weight β to the migration cost. This weight is chosen in proportion to the relative importance of migration cost, and it can be flexibly adjusted based on the preferences of MEC service provider.

TSP-VOP: Now, we present TSP-VOP:

$$\text{Minimize } \Gamma \tag{5}$$

Subject to:

$$\sum_{i\in\mathcal{M}} x_{ni}(t') = 1, \qquad \forall n, \forall t' \tag{6}$$

$$\sum_{i\in\mathcal{M}}\sum_{k\in\mathcal{A}} z_{nik}(t') = 1, \qquad \forall n, \forall t' \tag{7}$$

$$\sum_{k\in\mathcal{A}} z_{nik}(t') \le x_{ni}(t'), \qquad \forall n, \forall i, \forall t' \tag{8}$$

$$\sum_{i\in\mathcal{M}}\sum_{k\in\mathcal{A}} z_{nik}(t')\Lambda_{nik}(t') \le \mathbf{d}_n, \qquad \forall n, \forall t' \tag{9}$$

$$\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{A}} z_{nik}(t')E_{nik}(t') \le \mathbf{E}_i, \qquad \forall i, \forall t' \tag{10}$$

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}} y_{nij}(t+1) \le 1, \qquad \forall n \tag{11}$$

$$x_{ni}(t) + x_{nj}(t+1) - 1 \le y_{nij}(t+1), \forall n, \forall i, j, i \ne j \tag{12}$$

$$x_{ni}(t'), y_{nij}(t'), z_{nik}(t') \in \{0,1\}, \qquad \forall n, \forall i, j, \forall k, \forall t' \tag{13}$$

where $\forall t' \in \{t, t+1\}$. The objective function in Eq (5) represents an optimal cost that minimizes the sum of negative video coding ratios at the current time slot and the expected cost at the next time slot. Constraint (6) ensures each device is served by only one cloudlet. Constraint (7) ensures each device selects only one coding ratio for its video chunk each time. Constraint (8) is to ensure the coding ratio is selected only if a device is assigned to a cloudlet. Constraint (9) guarantees that the offloading delay does not exceed the deadline. Constraint (10) ensures total energy consumption of executing received video chunks in each cloudlet does not exceed its energy capacity. Constraint (11) guarantees the service migration happens for each device at most once between two consecutive time slots. Constraint (12) sets the migration decision variables according to whether a service migration happens or not. Constraint (13) guarantees that the decision variables are binary.

Let Ω be a set of all possible scenarios $\{\omega_1, \ldots, \omega_{|\Omega|}\}$, each of which has an associated probability p_s, where $s \in \{1, \ldots, |\Omega|\}$ represents an index of a scenario. The mathematical expectation $\mathbb{E}[W^{t+1}(X(t), Z(t), \xi(\omega))]$ can then be evaluated by:

$$\mathbb{E}[W^{t+1}(X(t), Z(t), \xi(\omega))] = \sum_{s\in|\Omega|} p_s W^{t+1}(X(t), Z(t), \xi(\omega_s))$$

As the number of possible scenarios grows exponentially with respect to the size of problem ($|\Omega| = \Psi^n$, where n is the number of mobile devices and Ψ is the number of possible location changes for each device), solving above equation is computationally intractable. To address this challenge, we propose an approximate data-driven solution presented next.

III. Data-Driven Video Offloading Algorithm

The basic ideal of SAA method is simple indeed: a sample h with specific size S (i.e., the number of scenarios) denoted by $\{\omega_1, \ldots, \omega_S\}$ is generated from the scenario set Ω according to probability distribution p. The mathematical expectation function $\mathbb{E}[W^{t+1}(X(t), Z(t), \xi(\omega))]$ is then approximated by the corresponding *sample average function*, meaning that:

$$\mathbb{E}[W^{t+1}(X(t), Z(t), \xi(\omega))] = \frac{1}{S}\sum_{s\in S} W^{t+1}(X(t), Z(t), \xi(\omega_s)) \tag{14}$$

Therefore, we introduce SAA-VOP, by formulating our TSP-VOP using SAA method. The objective function of SAA-VOP that corresponds to sample h is defined as follows:

$$\begin{aligned}\Gamma^{hS} = \min \sum_{n\in\mathcal{N}}\sum_{i\in\mathcal{M}}\sum_{k\in\mathcal{A}} -a_k\, z_{nik}(t) + \\ \frac{1}{S}\sum_{s\in S} W^{t+1}(X(t), Z(t), \xi(\omega_s)),\end{aligned} \tag{15}$$

where Γ^{hS} estimates the optimal cost for TSP-VOP (i.e., Γ). In addition, $X^{hS}(t)$ and $Z^{hS}(t)$ are the obtained offloading solutions (cloudlet and video quality decisions) for the decision variables of SAA-VOP that correspond to the solutions for our original TSP-VOP. The obtained SAA-VOP can be solved deterministically.

A. K-means Clustering for Scenario Reduction

In the traditional SAA method, a large number of scenarios for a sample are generated to estimate the expected function. To reduce the computational complexity of SAA and achieve approximate solutions, we propose a novel Clustering-based Sample Average Approximation Video Offloading Algorithm, called CSAA-VOA. Our approach utilizes K-means clustering to efficiently select fewer scenarios for each sample. Specifically, the generated S scenarios in each sample can be deemed as the observations in the K-means clustering, each of which is an N-dimensional vector, where N is the number of devices. A vector corresponding to a scenario $\omega_s \in \{\omega_1, \ldots, \omega_S\}$ can be denoted by $\vec{v}_s = ((l^x_{1,s}(t'), l^y_{1,s}(t')), \ldots, (l^x_{N,s}(t'), l^y_{N,s}(t')))$, where $(l^x_{n,s}(t'), l^y_{n,s}(t'))$ represents the two-dimensional coordinate of device $n \in \{1, \ldots, N\}$ at next time slot $t' = t+1$ under scenario ω_s.

A distance metric is required to calculate and compare the (dis)similarity between each pair of scenarios. Since each scenario is composed by two-dimensional coordinates of the devices, CSAA-VOA uses the Euclidean distance function for any two vectors $\vec{v}_a$ and $\vec{v}_b$ as: $f(\vec{v}_a, \vec{v}_b) =$

$$\sqrt{\sum_{n\in N}((l^x_{n,a}(t') - l^x_{n,b}(t'))^2 + (l^y_{n,a}(t') - l^y_{n,b}(t'))^2)}$$

CSAA-VOA utilizes K-means clustering to group all these S scenarios into C clusters based on this distance metric. A scenario which has the minimum distance to the centroid of its cluster is selected as the representative of that cluster.

A key property of CSAA-VOA is that it also takes into account the density of the clusters when calculating the sample average function. Since only one scenario is a representative of each cluster, computing the sample average function (i.e., Eq (14), which is the second term of the objective of SAA-VOP in Eq (15)) based on these centroid scenarios will deviate from the original expectation with all S scenarios and hence become ineffective. To tackle this challenge, CSAA-VOA introduces a density weight for each cluster. Considering π_c as the number of scenarios in cluster $c \in \{1, \ldots, C\}$, we assign a scaled weight as $\Pi_c = \frac{\pi_c}{S}$.

The total number of scenarios S for solving SAA-VOP now decreases to C, where $C \leq S$. The new set of scenarios is defined as $\{\bar{\omega}_1, \ldots, \bar{\omega}_C\} \subseteq \{\omega_1, \ldots, \omega_S\}$ and the associated weight set is denoted by $\{\Pi_1, \ldots, \Pi_C\}$. After clustering, we now reformulate the objective function of SAA-VOP, defined in Eq (15), based on the new set of scenarios as follows:

$$\hat{\Gamma}^{hC} = \min \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{M}} \sum_{k \in \mathcal{A}} -a_k \, z_{nik}(t) + \sum_{c \in C} \Pi_c \, W^{t+1}(X(t), Z(t), \xi(\bar{\omega}_c)), \tag{16}$$

where the sample size for solving SAA-VOP is reduced by utilizing K-means clustering. We refer to this as CSAA-VOP, Clustering-based SAA Video Offloading Problem.

B. Design of CSAA-VOA

We now present the details of our proposed Clustering-based SAA Video Offloading Algorithm, called CSAA-VOA, to solve TSP-VOP using SAA method. The implementation of our CSAA-VOA is summarized in Algorithm 1. CSAA-VOA will be executed at the beginning of each time slot.

At each current time slot t, CSAA-VOA generates a set of independent samples $\{1, \ldots, H\}$, each contains a large set of $\{\omega_1, \ldots, \omega_S\}$ scenarios for the new expected locations of the devices at next time slot $t+1$ according to the probability distribution (line 1). These samples are used to estimate the recourse cost function. An extra sample h' containing S' scenarios with $S' > C$ is generated in order to evaluate the obtained candidate solutions (line 2).

For each sample h, CSAA-VOA first utilizes K-means clustering to obtain a much fewer number of C scenarios, where $C \leq S$ (line 4). Then, the algorithm uses these filtered $\{\bar{\omega}_1, \ldots, \bar{\omega}_C\}$ scenarios to solve CSAA-VOP (Eq (16)) for that sample. CSAA-VOA computes the optimal value of the second stage for each device n, denoted as $W_s^{t+1}(x_{ni}(t), z_{nik}(t), \xi(\omega_s))$ using Eq (4) (lines 6). For each cloudlet i that is temporarily assigned to device n at current time slot t, CSAA-VOA finds $\hat{\Gamma}_{ni}^{hC}(t)$ based on the obtained W_s^{t+1} in all C scenarios of sample h using Eq (16) (lines 7). The above steps are repeated for each device in each sample h. Having all $\hat{\Gamma}_{ni}^{hC}(t)$ for all devices and cloudlets, CSAA-VOA needs to find the best allocation. This step can be modeled as a Generalized Assignment Problem (GAP) and be solved using existing approximation algorithms or heuristics (this is out of the scope of this study). To solve this assignment problem for each sample h, CSAA-VOA calls $Assign()$ function to compute the best candidate solution. This function returns the best offloading decision $X^{hC}(t)$ and $Z^{hC}(t)$ along with its cost Γ^{hC} (line 8). Therefore, for each of H samples, CSAA-VOA finds a candidate offloading solution.

Algorithm 1 CSAA-VOA

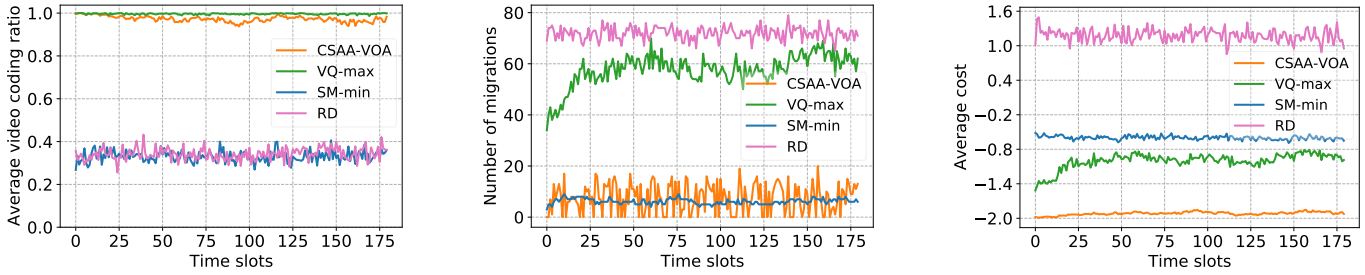
1: Generate H samples, each with S scenarios, where $S \geq C$
2: Generate an extra sample h' with S' scenarios, where $S' > C$
3: **for** each sample $h \in \{1, \ldots, H\}$ **do**
4: $\quad \{\bar{\omega}_1, \ldots, \bar{\omega}_C\} \leftarrow$ *K-means*$(C, \{\omega_1, \ldots, \omega_S\})$
5: $\quad$ **for** each device $n \in \mathcal{N}$ **do**
6: $\quad\quad$ **Calculate** $W_s^{t+1}(x_{ni}(t), z_{nik}(t), \xi(\omega_s))$
7: $\quad\quad$ **Calculate** $\hat{\Gamma}_{ni}^{hC}(t)$
8: $\quad \Gamma^{hC}, X^{hC}(t), Z^{hC}(t) =$ *Assign*$(\mathcal{N}, \mathcal{M}, < \hat{\Gamma}_{ni}^{hC}(t), \forall n, i >)$
9: **for** each sample $h \in \{1, \ldots, H\}$ **do**
10: $\quad$ **Assume** device n is offloading its video to cloudlet i
11: $\quad$ with coding ratio a_k at t based on obtained candidate
12: $\quad$ solution $(X^{hC}(t), Z^{hC}(t))$
13: $\quad$ **for** each device $n \in \mathcal{N}$ **do**
14: $\quad\quad$ **Calculate** $W_{s'}^{t+1}(x_{ni}(t), z_{nik}(t), \xi(\omega_{s'}))$
15: $\quad\quad$ **Calculate** $\hat{\Gamma}_{ni}^{hS'}(t)$
16: $\quad \Gamma^{hS'}, X^{hS'}(t), Z^{hS'}(t) =$ *Assign*$(\mathcal{N}, \mathcal{M}, < \hat{\Gamma}_{ni}^{hS'}(t), \forall n, i >)$
17: $\overline{\Gamma^{HS'}} \leftarrow \min_{h \in \mathcal{H}} \Gamma^{hS'}$
18: **return:** $\overline{\Gamma^{HS'}}$, $X^{hS'}(t), Z^{hS'}(t)$

Next, these candidate solutions are evaluated by an extra sample h' with size of S' scenarios (lines 9-16). Considering a candidate solution with $X^{hC}(t)$ and $Z^{hC}(t)$, the value of $W_{s'}^{t+1}$ for each scenario $\omega_{s'}$ is simply determined (line 14). Then, the best cost, $\hat{\Gamma}^{hS'}(t)$, is calculated by calling $Assign()$ function using all S' scenarios (line 16). The minimum value (denoted by $\overline{\Gamma^{HS'}}$) among all H values of $\hat{\Gamma}^{hS'}(t)$ is calculated (line 17). Finally, CSAA-VOA returns the minimum value as the best objective value and its associated solutions as the offloading decisions for devices at current time slot t (line 18).

IV. Performance Evaluation

A. Experimental Setup

Parameter Settings. We consider $M = 10$ cloudlets and $N = 80$ devices. The location of the cloudlets and devices are randomly chosen on a 2D grid. The length of a time slot is set to 300 milliseconds. The original size of each video chunk is uniformly selected from $[1, 10]$ MB and the deadline of each video chunk is set to less than the length of each time slot. To specify the required CPU cycles of a video chunk, we set $\kappa = 1000$ [12]. We consider 10 video coding ratios for each chunk as $\{0.1, 0.2, \ldots, 1\}$, where coding ratio of 1 means no compression. For each cloudlet i, its computing capabilities $\mathbf{C}_i$ is arbitrarily chosen from $[5, 25]$ GHz, bandwidth $\mathbf{B}_i$ is randomly selected from $[10, 30]$ MHz,



(a) Video quality (b) Service migration (c) Average cost

Fig. 1: Service performance under real-time movement trace

and the value of ϵ is set to 1.2×10^{-28} [13]. The energy consumption capacity of a cloudlet at each time slot is arbitrarily selected from $[8000, 15000]$ Joules [14]. As for the wireless communication, we set $p_0 = 0.5$ Watts and $N_0 = 1.0 \times 10^{-13}$ Watts. The migration cost coefficient is set $\beta = 5$ to highly reduce the number of migrations.

Scenario Generation. We use the real-world datasets of taxicab mobility traces in the central area of Rome [15] to capture the mobility traces of devices. The selected area with 4×4 km^2 is equally divided into a two-dimensional grid of more than 10×10 cells. We use this data to generate the scenarios for the devices based on the probability distribution of taxi movements. A device at the current time slot will either move to its neighboring cells at the next time slot or continue to stay in the current cell.

B. Effectiveness of CSAA-VOA

To evaluate the performance our CSAA-VOA under device mobility, we simulate a real-time movement trace and compare it with the following benchmarks:

- **Video quality maximization (VQ-max)**: At each time slot, each device is assigned to a cloudlet that maximizes the video coding ratio in order to offload the highest-quality video chunk.
- **Service migration minimization (SM-min)**: At each time slot, each device is assigned to a cloudlet that minimizes the migration cost in order to avoid a service migration.
- **Random (RD)**: At each time slot, each device is randomly assigned to a cloudlet to offload its video chunk.

To simulate the uncertain device mobility, a real-time movement trace for 180 consecutive time slots based on the scenario generator described in the setup is generated. We measure the service performance in terms of video quality, service migration, and the objective value (cost) for all 80 devices. Moreover, the sensitivity analysis of the cost is further evaluated. We set $H = 5$, $S = 100$ and $C = 10$ for CSAA-VOA.

Video Quality. Fig. 1a shows the average obtained video coding ratio per device. Obviously, VQ-max computes the optimal assignment of the video coding ratio (> 0.98) of the devices since it maximizes the video coding ratio. Both SM-min and RD result in values less than mean value of the video coding ratio. Our CSAA-VOA achieves significantly high average video coding ratio (> 0.90) compared with RD and SM-min (< 0.50).

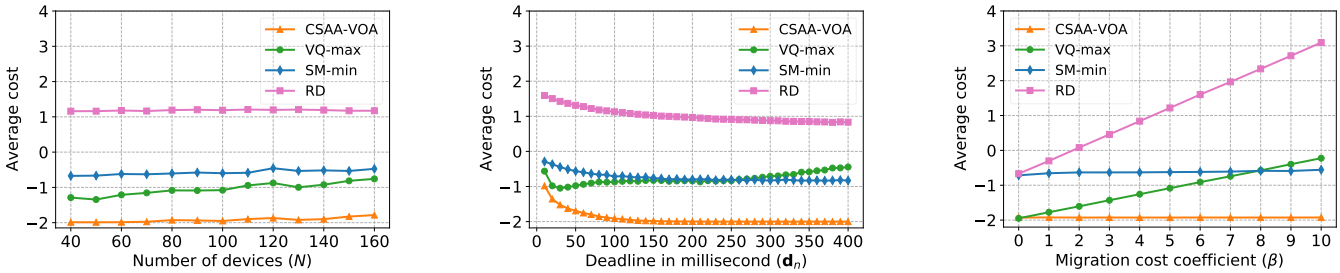
Service Migration. Fig. 1b shows required service migrations. This figure shows the number of migrations required for all devices. As expected, SM-min outperforms other approaches on overall in terms of minimum migration cost. RD performs the worst due to its random policy, while VQ-max also achieves a poor performance. Our CSAA-VOA leads to a much fewer service migrations compared with RD and VQ-max.

Objective Value (Cost). Fig. 1c shows a trade off between video quality and service migration. This figure shows the average cost $\bar{\Gamma}$ Eq (15) per device. Therefore, -2 represents the best (optimal) solution that all devices offload their video chunks with the highest quality (without compression using the video coding ratio of 1) while no migration cost is incurred. The results show that the proposed CSAA-VOA achieves near-optimal solutions. Specifically, CSAA-VOA outperforms VQ-max, SM-min, and RD by 23.16% to 53.97%, 60.25% to 73.45%, and 138.53% to 175.20%, respectively. This is due to the fact that CSAA-VOA effectively finds a trade off between the video coding ratio and the migration cost over time to compute the best offloading solution for the devices that leads to minimum migration cost and maximum video quality. The results also show that RD performs the worst due to its random policy.

Sensitivity Analysis of the Cost. We further investigate the impacts of some important parameters on the obtained cost. For a fair analysis, the average costs obtained by these approaches are evaluated using the same $S' = 200$ new scenarios.

The impact of number of devices is analyzed in Fig. 2a. Clearly, the performance of RD and SM-min are not greatly affected due to their random policy on selecting cloudlets for offloading. The average cost obtained by VQ-max increases as the number of devices increases. Because the deadline restriction (Constraint (9)) and the energy restriction (Constraint (10)) become tighter as the number of devices increases. Additionally, the average cost by the proposed CSAA-VOA remains the lowest.

The impact of the deadline for offloading a video chunk is shown in Fig. 2b. As deadline increases, the video quality



(a) under different N (b) under different deadline $\mathbf{d}_n$ (c) under different β

Fig. 2: Sensitivity analysis of average cost

of offloading can improve (by allowing a higher video coding ratio). Therefore, both RD and SM-min obtain better results (lower costs). Contrarily, the obtained average cost by VQ-max first decreases and then increases as the deadline increases. This is because the video coding ratio rapidly increases to reach a high value when deadline is small. However, the migration cost continues to increase. Differently, the average cost by CSAA-VOA monotonically decreases to the near-optimal cost as the deadline increases. This supports the fact that our CSAA-VOA adaptively computes the best offloading solution for devices by minimizing migration cost and maximizing the video coding ratio over time.

Fig. 2c shows the impact of the migration cost coefficient. When $\beta = 0$ (the migration cost is not considered), SM-min and RD obtain similar values while CSAA-VOA and VQ-max obtain similar values. Since at this case both SM-min and RD compute the offloading solution using their random policy. Contrarily, CSAA-VOA and VQ-max only optimize the video coding ratio. Further, when migration cost is considered (i.e., $\beta > 0$), the average cost by RD and VQ-max increases greatly as β increases. Because both approaches do not optimize the migration cost. However, the average cost obtained by SM-min is not impacted by β as it always minimizes the migration cost. Note that VQ-max will perform worse than SM-min when β is set to a large value. On the other hand, our proposed CSAA-VOA outperforms these benchmarks with stably lower average cost. This again supports the fact that CSAA-VOA effectively finds a trade off between the video coding ratio and the migration cost over time.

V. Conclusion

In this paper, we studied the video offloading problem in MEC to minimize migration cost and maximize video quality without a priori knowledge on device mobility. We formulated the problem as a two-stage stochastic program. Since our stochastic optimization problem is computationally intractable, we designed a clustering-based sample average approximation (SAA) method, called CSAA-VOA, to achieve an efficient scenario reduction while the quality of results is not negatively impacted. Through extensive experiments, the results have demonstrated the effectiveness of our proposed algorithm in video offloading.

Acknowledgment. This research was supported in part by Cisco grant CG#1935382.

References

[1] G. Forecast, "Cisco visual networking index: global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, 2019.

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[4] X. Chen, J.-N. Hwang, D. Meng, K.-H. Lee, R. L. de Queiroz, and F.-M. Yeh, "A quality-of-content-based joint source and channel coding for human detections in a mobile surveillance cloud," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 19–31, 2016.

[5] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia iot systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2017.

[6] L. Kong and R. Dai, "Efficient video encoding for automatic video analysis in distributed wireless surveillance systems," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 3, pp. 1–24, 2018.

[7] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[8] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[9] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Prof. of the IEEE Conference on Computer Communications*, 2019, pp. 1459–1467.

[10] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.

[11] E. Eriksson, G. Dán, and V. Fodor, "Predictive distributed visual analysis for video in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1743–1756, 2015.

[12] W. Ma and L. Mashayekhy, "Truthful computation offloading mechanisms for edge computing," in *Proc. of the IEEE International Conference on Edge Computing and Scalable Cloud*, 2020, pp. 199–206.

[13] W. Ma, X. Liu, and L. Mashayekhy, "A strategic game for task offloading among capacitated UAV-mounted cloudlets," in *Proc. of the IEEE International Congress on Internet of Things*, 2019, pp. 61–68.

[14] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: a stochastic optimization approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 909–922, 2019.

[15] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "CRAWDAD dataset roma/taxi (v. 2014-07-17)," Downloaded from https://crawdad.org/roma/taxi/20140717/taxicabs, Jul. 2014, traceset: taxicabs.