Delay-aware IoT Task Scheduling in Space-Air-Ground Integrated Network

Conghao Zhou*, Wen Wu*, Hongli He†, Peng Yang*, Feng Lyu*, Nan Cheng*, and Xuemin (Sherman) Shen*
*Department of Electrical & Computer Engineering, University of Waterloo, Canada
†College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China
{c89zhou, w77wu, p38yang, f2lyu, n5cheng, sshen}@uwaterloo.ca, hongli_he@zju.edu.cn

***Abstract*—Due to the versatile networking capability, space-air-ground integrated network (SAGIN) becomes a prominent future architecture to support the ever-increasing Internet of Things (IoT) applications. In this paper, we investigate the IoT task offloading under an SAGIN scenario where multiple IoT devices generate computing tasks to be processed. We adopt an unmanned aerial vehicle (UAV) to fly along a given trajectory to collect the tasks of IoT devices within the coverage area, and then makes the online offloading decision, i.e., processing locally, or offloading to the nearby base station or the far-away satellite. However, due to the constrained energy resources committed by UAV and the uncertainty of the system dynamics, designing an efficient computation task offloading algorithm is challenging. This dynamic scheduling problem is formulated as a constrained Markov decision process (CMDP), considering the stochastic channel conditions, UAV coverage, energy consumption, and task queue backlogs. By exploiting the stationary stochastic feature of the CMDP, the problem can be solved by the linear programming to find a stochastic policy. Simulation results demonstrate that the proposed computation offloading scheme can significantly reduce IoT task processing delay as compared to other benchmarks.**

***Index Terms*—Mobile edge computing, SAGIN, task offloading, IoT, MDP.**

I. Introduction

With the development of the Internet of Things (IoT), a soaring number of devices are expected to be connected, including wearable items, smart home appliances, industrial sensors, and autonomous vehicles [1], [2]. The limited computational capability and finite battery lives are challenging for IoT devices toward the realization of smart cities, smart grids, and industry. To further enhance the IoT services, mobile edge computing (MEC) has been proposed to offer high computational performance to individual IoT devices which typically have very limited energy and computational capability [3]. However, the MEC platform may be congested with the growing number of IoT devices if the computation resource is insufficient, and it may be unavailable in areas because of the lack of terrestrial network coverage. The space-air-ground integrated network (SAGIN) architecture is employed to enhance typical edge infrastructures (e.g., the base station (BS)) with satellite and aerial networks by providing full-coverage, flexible and efficient MEC services to IoT devices [1], [4]. Furthermore, under the SAGIN architecture, MEC can be available for IoT devices and reduce the cost of constructing new infrastructures with compromising functionalities to support different kinds of devices. Thus, SAGIN is a promising solution to provide adequate MEC resources and pervasive connections, so as to guarantee the required performance of IoT devices.

Most related works on MEC are based on terrestrial networks and focus on designing resource management and allocation schemes, including communication and computation resources, and designing the offloading policy for various computation tasks [3]. An energy-aware user-centric mobility management and distributed computation offloading algorithm are proposed in [5], [6], for a single-user system and a multi-user system, respectively. Instead of relying on full knowledge of network status information, Lyu *et al.* [7] proposed asymptotically optimal offloading schedules with partial out-of-date knowledge in the MEC. To keep the pace of the dynamic IoT computation offloading scenario, a learning-based approach is proposed in [8] in order to find the optimal offloading policy. Furthermore, a software-defined SAGIN architecture to manage resources and an efficient heterogeneous resource management via network slicing in SAGIN are proposed in [4] and [9], respectively. Considering dynamic user density and distribution, a 3-D drone-cell deployment scheme is proposed in [10] to cope with the coverage issues in SAGIN. Considering the coverage limitation of terrestrial network in some certain scenarios, computation offloading and resource allocation under SAGIN for IoT devices are studied in [1], where the dynamic environment features are captured by a learning-based approach. However, the proposed scheme in [1] works in a centralized manner to make decisions on computation task offloading, which calls for timely global information and a controller with high computation capability. Meanwhile, IoT devices are required to install three different communication interfaces for differentiated network connections, which could further raise the cost and energy consumption of IoT devices. Instead, in this paper, we use a UAV to collect computation tasks from IoT devices and make offloading decisions to reduce the cost of IoT devices, which also has low computational complexity without a global centralized controller.

In this paper, we investigate computation offloading of IoT applications in the SAGIN framework. To compensate the limited IoT coverage and capacity provided by BSs and satellites, a UAV is deployed as the "flying gateway" to collect IoT devices' tasks, and it can process computation tasks locally or decide the offloading strategy. Due to the uncertainty of IoT task arrivals, the mobility of the UAV, and the dynamic

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

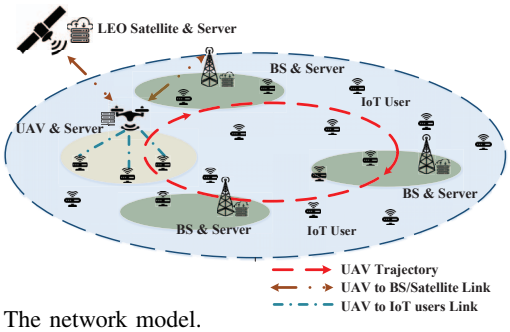

Fig. 1: The network model.

channel condition in SAGIN, it is difficult to find the optimal offloading strategy. To address this challenge, we formulate the problem as a Markov decision process (MDP). Considering the constraints of the UAV's buffer capacity and the energy consumption of computation and communication, a linear programming approach is employed to address this constrained Markov decision process (CMDP) problem. Compared with other heuristic schemes, simulation results demonstrate that our CMDP-based scheme is able to deliver advanced offloading decisions, which can minimize the long-term time averaged system delay, with taking into account the task drop rate and the UAV energy consumption.

The remainder of this paper is organized as follows. In Section II, the SAGIN architecture and computation offloading models are described. In Section III, problem formulation and transformation are provided, followed by a linear programming based solution. Simulation results are given to evaluate the performance of the proposed scheme in Section IV. We conclude this paper and direct our future work in Section V.

II. System Model

A. SAGIN Architecture

Consider an SAGIN architecture with N BSs, a UAV and a low earth orbit (LEO) satellite, where each entity is equipped with computing functionalities. Let $\mathcal{N} = \{0, 1, 2, \ldots, N\}$ be the set of indices of N BSs, and one LEO satellite indexed by 0. As shown in Fig. 1, a UAV is deployed to collect computation tasks from different IoT devices. Due to BSs' limited bandwidth for a large number of IoT devices, UAVs are considered as "flying gateway" for IoT devices [11]. As the coverage areas of both BS and UAV are limited, satellites installed with servers can act as the optional computation offloading solution for terrestrial cellular networks. We consider one LEO which has much smaller propagation delay, free space attenuation [12], and most commercial systems are using LEO, such as the Satrlink project by Space X. However, the UAV alone can only process a limited number of IoT tasks, and thus it needs to offload certain amount of tasks to the satellite or nearby BSs. Notice that we assume that a UAV flies along a scheduled trajectory, which is not the focus of our work. One can refer to [10] for the UAV trajectory design.

In this paper, we consider a discrete time-slotted system $\mathcal{T} = \{1, 2, \ldots, T\}$ with equal slot duration τ. The trajectory points of the UAV are denoted by $x_t = \{1, 2, \ldots, x_{\max}\}$, $t \in \mathcal{T}$, where $x_{\max}$ is the maximum number of locations. Different BSs can provide service to the UAV in the tth time slot. Denote $L_t \subseteq \mathcal{N}$ the set of available BSs during time slot t. We suppose that only one BS or satellite can be offloaded to in one time slot, and all offloaded tasks should be processed in the same edge server. Furthermore, the satellite or BS n can be connected by the UAV with a duration C_t^n (in time slots), $t \in \mathcal{T}$, where we assume the information of C_t^n is known *a priori* by the UAV according to the designed UAV trajectory. The amount of computation tasks from IoT devices to the UAV is different in each time slot. Denote by M_t, for $t \in \mathcal{T}$, the total amount of arrival tasks in tth time slot. M_t follows a general distribution, which can be acquired based on the statistical information of new arrival tasks collected from IoT devices.

B. Computation Task Model

The computation tasks collected from IoT devices by the UAV can be modeled as a tuple (ϕ, γ) [13], where $\phi \in [0, \phi_{\max}]$ (in bits) is the input data size of the computation task that needs to be offloaded, $\gamma \in [0, \gamma_{\max}]$ (in CPU cycles per bit) is the computation workload. $\phi_{\max}$ and $\gamma_{\max}$ are the maximum possible input data size and the maximum computation workload, respectively.

We use a_t, $t \in \mathcal{T}$, to represent the processing selection, which includes offloading to the satellite ($a_t = 0$) and offloading to a BS ($a_t = n, n \neq 0$). Note that the UAV makes decisions on whether perform local execution or edge offloading at the beginning of the time slot before new task arrival. The CPU frequency of the edge server installed on the satellite is denoted by f^0 (in CPU cycles per second). For accessible edge server on the BS $n \in L_t$, which has the CPU frequency f^n (in CPU cycles per second). As edge servers on BSs or the satellite need to provide services to many users and many tasks simultaneously, the available computation capability f^n may be different. Additionally, we assume the f^n does not change within different time slots. We denote by $m_t \in [0, m^{\max}]$ the amount of tasks offloaded to the edge servers on the satellite or BS n in the tth time slot, where $m^{\max}$ is the maximum amount of offloaded task, determined by UAV's residual energy. Thus, given the allocated CPU frequency, the computation delay of m_t tasks is

$$d_t^0(a_t, m_t) = \frac{m_t \phi \gamma}{f^n}, \ n \in L_t. \tag{1}$$

On the other hand, we consider the case of locally execution on the UAV. The local execution will undergo long queueing delay if excessive tasks are scheduled to be processed locally. Recall that the UAV processes tasks locally before it offloads to BSs or the satellite at the beginning of each time slot. Denote the CPU frequency of the edge server on the UAV by f^{uav} (in CPU cycles per second) and buffer capacity by ρ, respectively. Considering the delay of task delivery, the transmission delay of m_t tasks as H_t, which will be elaborated later. Thus, we calculate the amount of tasks v_t after local processing and offloading and the queue backlog of unaccomplished tasks q_t, $t \in \mathcal{T}$ by the end of time slot t

as follows

$$v_t = \max\left\{q_t - \left\lfloor \frac{f^{\text{uav}} H_t \tau}{\phi\gamma} \right\rfloor - m_t, 0\right\}, \tag{2}$$

$$q_t = \min\left\{v_{t-1} + M_{t-1},\, \rho\right\}, \tag{3}$$

where $\left\lfloor \frac{f^{\text{uav}} H_t \tau}{\phi\gamma} \right\rfloor$ is the amount of local execution tasks within one time slot, and $\lfloor\cdot\rfloor$ denotes the floor function.

If excessive tasks are executed locally, and the amount is larger than the buffer size ρ, newly collected tasks will be dropped. The amount of dropped tasks u_t, for $t \in \mathcal{T}$, can be expressed as follows

$$u_t = \max\left\{M_t + v_t - \rho, 0\right\}. \tag{4}$$

Notice that we consider the locally execution delay starts from the moment when the task is decided to process locally. Thus, the locally computation delay d_t^1 at the UAV can be calculated as follows

$$d_t^1(m_t) = \min\left(\frac{q_t \phi\gamma}{f^{\text{uav}}}, H_t\tau\right). \tag{5}$$

Additionally, the tasks which are neither processed locally nor offloaded need to wait in the queue, and we define the waiting time d_t^2 for these tasks as follows

$$d_t^2(m_t) = v_t H_t \tau. \tag{6}$$

C. Transmission Model

Suppose the UAV is equipped with two communication interfaces, one for communicating with the satellite and the other one for communicating with IoT devices and BSs. Since BSs and satellites use different spectrum to communicate, we do not consider the interference between BSs and the satellite in this paper. Meanwhile, we assume the UAV is sufficiently close to IoT devices, and thus the propagation delay from IoT devices to the UAV is neglected [14].

Adopting the link path loss model between the UAV and BS in [10], the average UAV to BS path loss, for $n \neq 0$ can be calculated as follows

$$\text{PL}\left(x, \theta\right) = 10\alpha \log\left(x\right) + B\left(\theta - \theta_0\right) e^{\frac{\theta_0 - \theta}{C}} + \eta_0, \tag{7}$$

where x indicates the horizontal distance between the UAV to a BS, and θ is the vertical angle between the UAV and a BS. Additionally, the terrestrial pathloss exponent, angle offset, excess path loss scalar, angle scalar and excess path loss offset are represented by α, θ_0, B, C and η_0, respectively. Since the UAV moves along its trajectory, the distance and angle between the UAV and a BS vary over different time slots. Based on (7), the uplink transmission rate from the UAV to BS n, for $n \neq 0$, at time slot t is given by

$$r_t^n = W_{\text{B}} \log_2\left(1 + \frac{P_{\text{B}} \cdot 10^{\frac{\text{PL}}{10}}}{\sigma_{\text{B}}^2}\right), \tag{8}$$

where W_{B} indicates the channel bandwidth of the link between the UAV and a BS, P_{B} is the transmit power from the UAV to a BS, and σ_{B}^2 is the power of the background noise.

Alternatively, when $a_t = 0$, it indicates that tasks are offloaded to the satellite. We adopt the Weibull-based channel model in [15] to generate the rain attenuation for terrestrial to satellite links. Ignoring the distance variation between the UAV and the satellite, the channel can be assumed to be static along the UAV trajectory. Then the channel gain between the UAV and the satellite is given as

$$h = \frac{G_{\text{tx}} G_{\text{rx}} \lambda^2}{\left(4\pi x^{\text{sat}}\right)^2} 10^{-\frac{F_{\text{rain}}}{10}}, \tag{9}$$

where x^{sat} is the distance between the UAV and the satellite, G_{tx} and G_{rx} are antenna gains of the UAV and the satellite, respectively. The distribution of rain attenuation F_{rain} follows the Weibull distribution [15]. Given the channel gain, we can calculate the transmission rate of the UAV to satellite link as follows

$$r_t^0 = W_{\text{S}} \log_2\left(1 + \frac{P_{\text{S}} \cdot |h|^2}{\sigma_{\text{S}}^2}\right), \tag{10}$$

where W_{S} is the channel bandwidth of the link between the UAV and the satellite, σ_{S}^2 is the power of the background noise, and P_{S} indicates the UAV transmit power to the satellite.

As the time duration that the UAV can stay connected to a BS is limited, we should guarantee the number of transmission slots is smaller than the maximum number of connected slots C_t^n, i.e., $H_t \leqslant C_t$. The UAV needs to finish transmitting m_t tasks within H_t time slots, which is given by

$$H_t = \arg\min_k \left(\sum_{i=t}^{t+k} r_i^n \tau \geqslant m_t \phi\right). \tag{11}$$

If the UAV offloads some tasks to BS n, it cannot offload new tasks to any BS or satellite in later H_t time slots. The UAV needs to process new task arrivals locally until offloaded tasks transmission are accomplished. Considering the long distance between the UAV and the satellite, we denote the propagation delay from the UAV to the satellite by d^{sat}. The delay for transmitting tasks from the UAV to an edge server on BS n or the satellite, is given by

$$d_t^3(a_t, m_t) = \begin{cases} \dfrac{H_t\tau}{m_t} + d^{\text{sat}}, & a_t = 0 \\ \dfrac{H_t\tau}{m_t}, & a_t = n, n \in L_t, n \neq 0. \end{cases} \tag{12}$$

D. Energy Consumption Model

If the UAV trajectory and UAV parameters are fixed, we can consider that the propulsion energy is a constant [16]. We strive to ensure the energy consumption of the remaining components do not exceed the budget, which contains both the communication-related and computing-related energy. Denote the expectation of energy consumption budget by ε, which is imposed to the communication-related and computing-related energy activities.

The main part of the UAV communication-related energy is the transmission energy, which is the product of the transmit power and the transmission time. Denote the transmit power from the UAV to BSs and the satellite by P_{S} and P_{B},

respectively. Then the transmission energy, including BSs and satellite can be given as

$$e_t^{\mathrm{o}}(a_t, m_t) = \begin{cases} P_{\mathrm{S}} \cdot d_t^3, & a_t = 0 \\ P_{\mathrm{B}} \cdot d_t^3, & a_t = n, n \neq 0. \end{cases} \tag{13}$$

The computing-related energy is mainly caused by the local execution. The CPU energy can be calculated based on the CPU frequency and the computation workload. Thus, the local execution energy e_t^{l}, for $t \in \mathcal{T}$, is expressed as

$$e_t^{\mathrm{l}} = \min\left\{q_t \phi \gamma, f^{\mathrm{uav}} H_t \tau\right\} \cdot \xi \left(f^{\mathrm{uav}}\right)^2, \tag{14}$$

where ξ indicates the effective switched capacitance determined by the chip architecture [17].

III. Problem Formulation

As the link availability and task arrival are highly dynamic and uncertain, and the location and the task backlog evolve in an ergodic way, we concentrate on minimizing the long-term average system delay. Therefore, to address this problem, the stationary decisions are adopted, which are time-invariant and just related to the current system status. Due to the optimality of the stationary decision in the ergodic system, this problem can be formulated as an MDP to find a stationary decision [18].

A general MDP problem consists of five parts, which includes the state space denoted by $\mathcal{S}$, the decision space denoted by $\mathcal{A}$, state transition probabilities $P := \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, the reward function $R := \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the policy $\mathbf{\Pi}$, which is a decision rule mapping from a state to an action.

A tuple denoted by $\boldsymbol{s} = (x, q), \boldsymbol{s} \in \mathcal{S}$ is used to describe the system state, where x and q are the UAV location and current queue backlog, respectively. A decision is made based on the current state, and the decision is denoted by a tuple $\boldsymbol{a} = (a, m), \boldsymbol{a} \in \mathcal{A}$, where a is used to indicate offloading decision to which BS or satellite, and m denotes the amount of offloaded tasks.

The state transition includes two parts: the trajectory point x, which is only updated along the known UAV trajectory; the queue backlog in the UAV buffer. We use $f(\cdot)$ and $F(\cdot)$ to define the probability mass function and cumulative distribution function of new arrival task amount M, respectively. Thus, the probability of the state transition is denoted by $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ which can be derived as follows

$$p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} f(H(q' - v)), & \text{case 1;} \\ 1 - F(H(q' - v)), & \text{case 2;} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where case 1 indicates $x' = x + H, v \leq q' \leq \rho - 1$; case 2 indicates $x' = x + H, q' = \rho$.

Then reward function $r(\boldsymbol{s}, \boldsymbol{a})$ on state $\boldsymbol{s}$ with action $\boldsymbol{a}$ is defined as follows

$$r(\boldsymbol{s}, \boldsymbol{a}) = -\left(d^0 + d^1 + d^2 + d^3\right), \tag{16}$$

where d^0 is the computation delay of offloaded tasks, d^1 means the computation delay of tasks locally processed, d^2 indicates the waiting time of tasks not offloaded or not processed locally, and d^3 is the transmitting delay of offloaded tasks.

We denote the stationary deterministic policy by $\mathbf{\Pi}$, which means that each state $\boldsymbol{s}$ is assigned with a fixed action $\boldsymbol{a}$ and this action will be always be used whenever the system is in this state. Note that the accessible action $\boldsymbol{a}$ should satisfy $H_t \leqslant C_t$, where $a \in \{0, 1, \ldots, N\}$ and $m \in \{0, 1, \ldots, m^{\max}\}$.

The expectation of long-term average communication-related and computing-related energy consumption is given by

$$\lim_{T \to \infty} \mathbb{E}\left[\frac{\sum_{t=1}^{T} \left(e_t^{\mathrm{o}} + e_t^{\mathrm{l}}\right)}{T} \middle| \mathbf{\Pi}\right] \leqslant \varepsilon, \tag{17}$$

where the expectation $\mathbb{E}[\cdot]$ is taken.

To avoid excessive task drop, we need to guarantee the expectation of the long-term drop rate is lower than a given threshold η,

$$\lim_{T \to \infty} \frac{\mathbb{E}\left[\sum_{t=1}^{T} u_t \middle| \mathbf{\Pi}\right]}{\mathbb{E}\left[\sum_{t=1}^{T} M_t \middle| \mathbf{\Pi}\right]} \leqslant \eta, \tag{18}$$

where η depends on the IoT service requirements.

Thus, the computation offloading problem can be formulated as the following optimization problem in MDP, i.e.,

$$\max_{\mathbf{\Pi}} \lim_{T \to \infty} \mathbb{E}\left[\frac{\sum_{t=1}^{T} r(\boldsymbol{s}, \boldsymbol{a})}{T} \middle| \mathbf{\Pi}\right] \tag{19a}$$

$$\text{s.t. } (17), (18). \tag{19b}$$

The aforementioned problem is an extension of MDP problem, which is with some constraints. Generally, iterative methods, such as the policy iteration and value iteration, can find a deterministic policy to solve the MDP problem. However, they are unsuitable for finding a deterministic policy in the CMDP problem, which dictates what specific action to take given a particular state. The reason is that both the policy iteration and value iteration are value-based framework, which are difficult to be compatible to the MDP problem with extra constraints [19]. Instead, assuming the underlying process is ergodic and stationary, we can find a stochastic policy to solve this problem, which is a probability distribution of mapping from a state $\boldsymbol{s}$ to an action $\boldsymbol{a}$. The stochastic policy is the relaxation form of the deterministic policy, and the solution of the stochastic policy can act as the upper bound of the deterministic policy [18]. Thus, we need to reformulate the original problem to find the stochastic policy for this CMDP problem.

We denote the stochastic policy by $\boldsymbol{\pi}$, which includes all steady-state probabilities $\pi(\boldsymbol{s}, \boldsymbol{a})$ for every accessible state-action pair. Thus, the original problem can be reformulated to

TABLE I: Simulation Parameters

Parameter	Value	Parameter	Value
N	4	f^{uav}	3 GC/s
ϕ	10 MB	f^0	5 GC/s
γ	25 cycles/bits	$f^n, n \neq 0$	10 GC/s
W_{B}	10 MHz	N_0	-174 dBm/Hz
W_{S}	5 MHz	P_{B}	1.6 W
P_{S}	5 W	ξ	10^{-27}
d^{sat}	6.44 ms	$m^{\max}$	8
ρ	20		

find a stochastic policy $\boldsymbol{\pi}$ as follows

$$\max_{\boldsymbol{\pi}} \sum_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}} r(\boldsymbol{s},\boldsymbol{a})\pi(\boldsymbol{s},\boldsymbol{a}) \tag{20a}$$

$$\text{s.t.} \sum_{\boldsymbol{a}\in\mathcal{A}} \pi(\boldsymbol{s}',\boldsymbol{a}) = \sum_{\boldsymbol{s}\in\mathcal{S}}\sum_{\boldsymbol{a}\in\mathcal{A}} p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\pi(\boldsymbol{s},\boldsymbol{a}) \quad \forall \boldsymbol{s}' \tag{20b}$$

$$\sum_{\boldsymbol{s}\in\mathcal{S}} \pi(\boldsymbol{s},\boldsymbol{a}) = 1 \tag{20c}$$

$$\frac{\sum_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}} u(\boldsymbol{s},\boldsymbol{a})\pi(\boldsymbol{s},\boldsymbol{a})}{\sum_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}} M(\boldsymbol{s},\boldsymbol{a})\pi(\boldsymbol{s},\boldsymbol{a})} \leqslant \eta, \tag{20d}$$

$$\sum_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}} (e^{\text{o}}(\boldsymbol{s},\boldsymbol{a}) + e^{\text{l}}(\boldsymbol{s},\boldsymbol{a}))\pi(\boldsymbol{s},\boldsymbol{a}) \leqslant \varepsilon, \tag{20e}$$

where (20a) is the objective function to be maximized, (20b) represents the global equilibrium equation of the steady-state probabilities, (20c) describes the fact that the sum of all the policy probabilities must be 1, (20d) shows that the expectation of the task drop rate should be guaranteed, and (20e) gives the limitation of the expectation of average communication-related and computing-related energy consumption.

As the system is assumed to be ergodic and stationary, the expectation of long-term task drop rate and average energy consumption are also policy-related. Thus, the energy consumption in (17) and the task drop rate in (18) can be reformulated, respectively. By reformulating the original problem, we can adopt the linear programming method to address this CMDP problem by finding a stochastic policy. There are off-the-shelf algorithms for solving linear programming problem in MDP that are of polynomial time complexity [20]. Therefore, this CMDP problem can be solved within polynomial time of $|\mathcal{S}|$ and $|\mathcal{A}|$.

IV. Performance Evaluation

In this section, the MATLAB-based simulation results are carried out to evaluate the proposed offloading strategy. In our simulation, a single UAV flies along a fixed circular trajectory with 500 m radius, and 100 m altitude. We consider the LEO satellite connection is always available. The amount of new arrival tasks M_t is assumed to follow the Poisson distribution with arrival rate μ. For the suburban environment, $(\alpha, \theta_0, B, C, \eta_0)$ is given as $(3.04, -3.61, -23.29, 4.14, 20.7)$ [10]. For the rain attenuation of the satellite channel model, F_{rain} is set to 6 dB. Other simulation parameters are listed in Table I.

Figure 2 shows the delay of the system, where the energy consumption increases from 20 W*sec to 30 W*sec. In the experiment, the drop rate $\eta = 0.05$, and the task arrival rate is set with $\mu = 2, 2.3, 2.5$, respectively. Fig. 2 shows that the system delay can be reduced by providing more energy, since more tasks can be offloaded to BSs or the satellite. In addition, the total tasks' processing delay increases with the amount of task arrivals, when the energy budget is fixed. However, due to the amount of maximum offloaded tasks, the system processing capability is limited. Even more consumption energy is allowed, it cannot process more tasks.

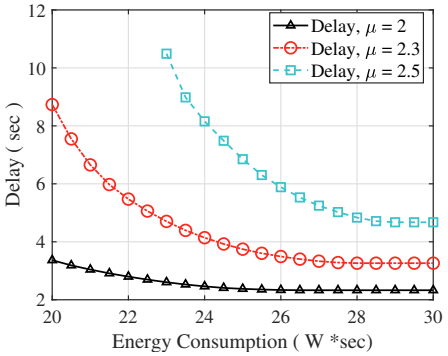

Fig. 2: System delay versus energy consumption in different task arrival rates μ, where task drop rate η = 0.05.

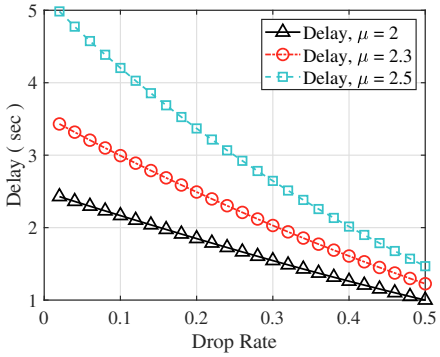

Fig. 3: System delay versus system drop rate in different task arrival rates μ, where energy budget ε = 35 W*sec.

In Fig. 3, we show the impact of the drop rate on the system delay, where task arrival rates are set with $\mu = 2, 2.3, 2.5$, and the energy budget is set with $\varepsilon = 35$ W*sec. System delay decreases as the required task drop rate grows, as more computation tasks can be dropped. As expected, the system delay increases as the growth of the task number when the required task drop rate is fixed.

Figure 4 shows the system delay under the proposed scheme, the random scheme and the greedy scheme by varying task arrival rates μ from 2 to 4. The random scheme means that the accessible actions are given by a random probability in different states. The greedy scheme selects BS prior to offload more tasks, and if the UAV cannot select a BS, it will offload more tasks to the satellite. In the low arrival rate case, the greedy scheme is better than random scheme, but it is the worst scheme in the high arrival rate cases, because the amount of new task arrivals may be more than that of the offloaded tasks. Offloading excessive tasks at every state will reduce the amount of tasks processed by system. Compared

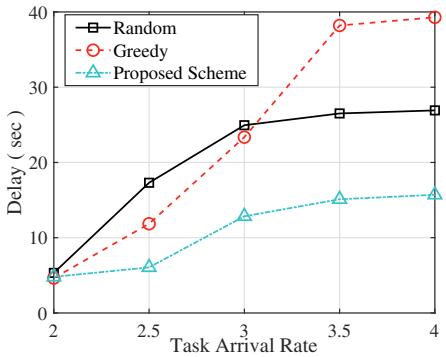

Fig. 4: Comparison of the system delay versus different arrival rates μ.

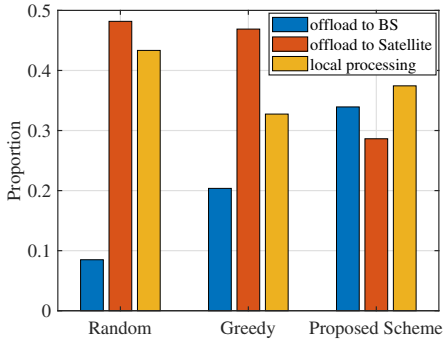

Fig. 5: Proportion of the tasks offloaded to BS, offloaded to satellite or processed locally, where μ = 2.5.

with other two schemes, the proposed scheme has the lowest system delay, and it is applicable to different arrival rates.

Finally, Fig. 5 shows the offloading proportion under different schemes with the arrival rate $\mu = 2.5$. The proposed scheme can make the UAV offload certain tasks to BSs when they are covered by BS, and offload to the satellite when it is out of the BS coverage area. Offloading to the satellite is an important backup for BSs, which effectively reduces task drop rate. Other schemes can select the number of offloading tasks, but they cannot ensure the energy and task drop rate simultaneously.

V. Conclusion

In this paper, we have investigated the computation offloading problem under an SAGIN scenario. Both the UAV's data buffer capacity and energy budget for the computation and communication on the flight are considered in this problem. By modeling the system state as the UAV location distribution and the task backlog in the buffer, our scheme is able to make decisions on the channel access and task offloading by solving a CMDP problem, which aims to minimize the expectation of long-term average system delay with balancing the system task drop rate and average energy consumption in the long term. Simulation results have demonstrated the effectiveness of the proposed SAGIN computation offloading scheme. For the future work, MEC with an unknown distribution of task arrival and unfixed UAV trajectory will be investigated to further enhance the robust performance of the proposed scheme.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 91638204 and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[2] F. Lyu, H. Zhu, N. Cheng, H. Zhou, W. Xu, M. Li, and X. Shen, "Characterizing urban Vehicle-to-Vehicle communications for reliable safety applications," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–17, Jun. 2019.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[4] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101–109, Jul. 2017.

[5] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Oct. 2017.

[6] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

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

[8] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Reinforcement learning-based computing and transmission scheduling for LTE-U-enabled IoT," in *Proc. IEEE Globecom*, Abu Dhabi, UAE, Dec. 2018.

[9] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, "Air-ground integrated vehicular network slicing with content pushing and caching," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2114–2127, Aug. 2018.

[10] W. Shi, J. Li, N. Cheng, F. Lyu, S. Zhang, H. Zhou, and X. Shen, "Multi-drone 3-D trajectory planning and scheduling in drone-assisted radio access networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8145–8158, Aug. 2019.

[11] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, May 2016.

[12] F. Vatalaro, G. E. Corazza, C. Caini, and C. Ferrarelli, "Analysis of LEO, MEO, and GEO global mobile satellite systems in the presence of interference and fading," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 2, pp. 291–300, Feb. 1995.

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Survey Tuts.*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[14] N. Hosseini, H. Jamal, J. Haque, T. Magesacher, and D. W. Matolak, "UAV command and control, navigation and surveillance: A review of potential 5G and satellite systems," pp. 1–10, March 2019.

[15] S. A. Kanellopoulos, C. I. Kourogiorgas, A. D. Panagopoulos, S. N. Livieratos, and G. E. Chatzarakis, "Channel model for satellite communication links above 10GHz based on Weibull distribution," *IEEE Commun. Lett.*, vol. 18, no. 4, pp. 568–571, Feb. 2014.

[16] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Mar. 2017.

[17] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," pp. 1–11, Feb. 2019.

[18] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[19] H. He, H. Shan, A. Huang, and L. Sun, "Resource allocation for video streaming in heterogeneous cognitive vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7917–7930, Mar. 2016.

[20] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. ACM STOC*, Washington, USA, Apr. 1984.