Optimal Edge Computing for Infrastructure-Assisted UAV Systems

Davide Callegaro, *Member, IEEE,* and Marco Levorato, *Member, IEEE,*

***Abstract*—The ability of Unmanned Aerial Vehicles (UAV) to autonomously operate is constrained by the severe limitations of their on-board resources. The limited processing capacity and energy storage of these devices inevitably makes the real-time analysis of complex signals – the key to autonomy – challenging. In urban environments, the UAVs can leverage the communication and computation resources of the surrounding city-wide Internet of Things infrastructure to enhance their capabilities. For instance, the UAVs can interconnect with edge computing resources and offload computation tasks to improve response time to sensor input and reduce energy consumption. However, the complexity of the urban topology and large number of devices and data streams competing for the same network and computation resources create an extremely dynamic environment, where poor channel conditions and edge server congestion may penalize the performance of task offloading. This paper develops a framework enabling optimal offloading decisions as a function of network and computation load parameters and current state. The optimization is formulated as an optimal stopping time problem over a semi-Markov process. We solve the optimization problem using Dynamic Programming and Deep Reinforcement learning at different levels of abstraction and prior knowledge of the system underlying stochastic processes. We validate our results in a realistic scenario, where a UAV performs a building inspection task while connected to an edge server.**

***Index Terms*—Edge Computing, Urban Internet of Things, Unmanned Aerial Vehicles, Autonomous Systems.**

I. Introduction

The use of Unmanned Aerial Vehicles (UAV) is being increasingly proposed for a wide spectrum of applications, including surveillance and monitoring, disaster management, agriculture, and network coverage extension [1]–[5]. Their autonomous operations require the acquisition and real-time analysis of information from the surrounding environment. However, processing information-rich signals, such as video and audio input, to inform navigation and, in general, autonomous decision making, is an extremely demanding task for these constrained platforms. In fact, due to the limited on-board computation resources, the analysis process may require a significant amount of time, thus decreasing the UAV responsiveness to stimuli. Additionally, continuously running heavy-duty analysis algorithms imposes a considerable energy expense burden to these battery-powered devices. In summary, the degree of autonomy of UAVs may be limited, and may come at the price of a reduced operational lifetime.

D. Callegaro and M. Levorato are with the Department of Computer Science, University of California, Irvine, CA, 92697.
E-mail: {dcallega, levorato}@uci.edu

This work was partially supported by the NSF under grant IIS-172433.

In urban environments, the UAVs can leverage the resources of the surrounding urban Internet of Things (IoT) infrastructure to overcome some of its limitations and enhance its capabilities. For instance, the UAV can use the communication infrastructure to connect to edge servers – that is, compute-capable machines positioned at the network edge – to offload data processing tasks [6]. By offloading the computation task to a more powerful device, a UAV can possibly reduce the capture-to-decision time, thus improving its reaction time to sensor input. Additionally, offloading data processing to an edge server can reduce the amount of energy needed to complete the mission.

However, the urban IoT is a highly dynamic system, where a myriad of devices, and data streams, compete for the available communication and computation resources. As a result, the network connecting the UAV and edge server may be congested, and the transportation of the data to the edge server may require a large time. Additionally, the topology of urban areas may degrade the capacity of the channel due to path loss and shadowing. Finally, the edge server may have a queue of computation tasks from other devices and services that need to be completed before processing the data from the UAV. Therefore, in some conditions and locations, the time needed to transport the data to the edge server and receive the outcome of analysis may exceed that of local processing at the UAV.

In this paper, we present an optimized decision process through which the UAV decides whether to process locally or offload the computation task to the edge server. The decision is based on a series of interactions between the UAV and the IoT system, where the UAV receives feedback on the state of the network and edge server, which allows the estimation of the residual time to task completion. Based on this information, the UAV solves an optimization problem aiming at the minimization of a weighted sum of delay and energy expense. Formally, the problem is formulated as an Optimal Stopping Time problem over a semi-Markov process.

Numerical results, which are based on parameters extracted from a real-world implementation of the system, demonstrate that the proposed intelligent and sequential probing technique effectively adapts the processing strategy to the instantaneous state of the network-edge server system. The outcome is a reduced processing delay and energy expense, two extremely important metrics in the considered application. These results are evaluated on the aforementioned urban scenario, where we place particular emphasis on the components that could decrease the performance of the UAV across a mission. In addition to analytical evaluations, we characterize the, temporal

and average, performance of the adaptive offloading scheme we proposed in a scenario where a UAV mission requires to complete a trajectory around a building while analyzing images.

In the setting described above, we train a Deep Reinforcement Learning (DRL) agent capable of learning spatio-temporal characteristics of the environment to learn effective offloading policies. Results indicate the importance of features such as position and recent offloading outcomes in maximizing the ability of the controller to optimize its decisions across missions.

The rest of the paper is organized as follows. Section II provides an overview of the system considered in this paper. Section III describes in detail the parameters and operations of the UAV-edge server system. Section IV introduces a Markovian description of the system's dynamics and formulates and solves the problem for the optimization of the offloading decisions. Section V presents numerical results illustrating the performance of the proposed adaptive offloading strategy, and comparing it with alternative solutions. Section VI discuss related research and Section VII concludes the paper.

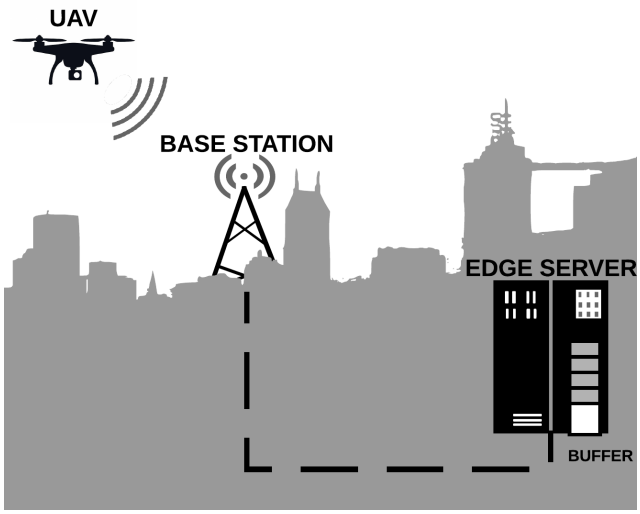


Figure 1: Illustration of the considered scenario and system: a UAV interconnects with an edge server through a low latency wireless link to offload computation tasks. Poor channel conditions and high processing load at the edge server may result in a larger delay and energy expense compared to local on-board processing.

II. System and Problem Overview

We consider a scenario where a UAV autonomously navigates an urban environment. The UAV is assigned the task to acquire and process complex data in predefined locations within the city, where the outcome of processing may influence sensing and navigation actions. A relevant case-study application is city-monitoring, in which the UAV captures a panoramic sequence of pictures at each location and process them using a classification algorithm to detect objects or situations of interest. In case of positive detection, the UAV may stay at the location to capture more detailed or higher-resolution pictures of a specific portion of its view.

Intuitively, processing information-rich signals using a computation-intense algorithm is a challenging task for inherently constrained platforms such as UAVs. In fact, the limited processing power of the on-board computation resources results in long capture-to-output time of the algorithm, which decreases responsiveness to stimuli and increases mission time. Additionally, on-board processing consumes a significant amount of energy, even when compared to motion and navigation, thus shortening the lifetime of these battery-powered systems. Note that in the scenario described above, the UAV is hovering while waiting for the classification algorithm to complete, as the outcome will determine its subsequent action. Thus, a large processing time incurs at additional energy expense penalty associated with longer flight time.

The UAV can leverage the resources of the surrounding urban IoT infrastructure to improve its performance. In the scenario at hand, edge servers placed at the network edge can take over the task of processing the data acquired by the UAV. Intuitively, the larger processing power of edge servers compared to that of UAVs grants a much faster completion of the processing task, thus allowing a faster decision making and a smaller capture-to-decision time, defined as the time between image capture and the availability of the its analysis' outcome. Additionally, the UAV would be relieved from the energy expense burden of processing, at the price of energy expense associated with data transmission. We remark that a shorter time to receive the output of the classification algorithm also corresponds to a smaller energy expense associated with hovering.

However, as noted in the introduction, the urban IoT is a highly dynamic environment, where a myriad of data streams and services coexist and compete for the same communication resources. In the considered scenario, the wireless channel connecting the UAV to a wireless access point may have a low capacity due to the physical properties of signal propagation, but also due to the existence of interfering communications which use part of the time/frequency channel resource. Additionally, the edge servers may be serving other devices offloading their computation tasks, and the UAV task may suffer queuing delay, or a reduced processing speed. As a result, in certain conditions, offloading the computation task to an overloaded edge server connected to the UAV through a poor communication channel may lead to a longer capture-to-decision time. Again, this corresponds to less efficient mission operations, but also a possibly large energy expense due to hovering while waiting for a response.

In order to fully harness the possible performance gain granted by the available resources provided by the urban IoT infrastructure, the UAV needs to make informed decisions about whether or not to offload the execution of image analysis. To this aim, we equip the UAV with the ability to interact with the surrounding network and edge devices and acquire information regarding the status of the communication and processing pipeline. The information is used to evaluate the progress of the task and predict the future cost of the

binary decision between local and edge-assisted computing.

We remark that the optimization framework can be used in scenarios with multiple base stations and edge servers. As the sequential decisions are made on a task by task basis, handover and connectivity can be managed by the network infrastructure without a major impact on decision making. However, we note that when considering agents learning spatio-temporal correlation properties (as those shown in Section V.C), the agent will need to implicitly incorporate connectivity information associated with spatial features. In fact, handover may cause abrupt loss of temporal correlation – for instance in server load if a different edge server is connected to the new base station.

III. System Model

In this section, we formalize and discuss an abstraction of the system composed of the UAV, a network access point and an edge server. We divide the description into modules focusing on the communication, computation and energy expense aspects of the system.

A. Communications

The UAV is connected to the network access point through a wireless channel of finite capacity. The data to be transferred for offloading have size L-bits. The UAV transmits with fixed power P and rate R in the finite set of $K+1$ transmission rates $\{R_0, R_1, R_2, \ldots, R_K\}$, where $R_0{=}0$ corresponds to disconnection from the network, and thus no data transmission. The link between the UAV and the AP is a wireless link affected by path loss, fading and noise. The SNR at the receiver is

$$\mathrm{SNR} = \frac{gP}{\sigma^2}, \tag{1}$$

where σ^2 is the noise power and g is the channel attenuation coefficient including path loss and fading. We assume exponential path loss and Rayleigh flat fading. Thus, the distribution of g is

$$\Theta_g(x){=}\Pr(g \leq x){=}1{-}e^{-\frac{x}{\gamma}}, \tag{2}$$

where γ is the path loss.

Assuming channel knowledge and a capacity achieving scheme, the selected transmission rate of the UAV is equal to R_i bits/s if $g \in (g_i, g_{i+1}]$, where

$$g_i = g : R_i = \mathcal{C}(g\mathrm{SNR}), \quad i{=}1, \ldots, K, \tag{3}$$

and

$$\mathcal{C}(x){=}\log(1+x). \tag{4}$$

The resulting transmission time is L/R_i seconds. In the paper, we use a capacity model to abstract the communication layer, where the channel gain is matched with a maximum achievable data rate. The integration in the model of more realistic communication models, for instance to capture interactions between physical, channel access and transport layers, would lead to a much more convoluted analysis. We point the interested reader to our work [7] for the evaluation and analysis of real-world edge computing for UAVs with dynamic offloading.

B. Computation

The time to complete the computation task locally at the UAV and at the edge server are captured using the random variables X' and X, respectively. We assume that X' and X follow an exponential distribution of rate μ' and μ tasks/s, respectively. The edge server accumulates incoming computation tasks in a finite buffer of size B tasks. Excluding the task generated by the UAV, tasks arrive according to a Poisson process of rate λ tasks/s, with $\lambda{<}\mu$.

C. Energy

As described in the previous section, at each predefined location the UAV captures the data, and then completes the computation task – either locally or at the edge server – while hovering maintaining the position. We define a rate of energy expense for the two fundamental operational blocks that are influenced by the offloading decision: processing and hovering. Specifically, we define P_{P} and P_{H} as the Watts used to respectively process the data and hover. As mentioned earlier, the transmission power is equal to P Watts.

IV. Optimal Offloading Decisions

In the considered scenario, the two most relevant performance metrics are energy expense E and time T per location. Herein, we assume the state of the system at each location to be independent. Importantly, the costs E and T are a function of the offloading decision, that is, whether the computation task is completed at the UAV or at the edge server.

Given the knowledge of the system parameters, the UAV can compute the average cost E and time T corresponding to each of the two options, where the average is over realizations of the stochastic process associated with the system dynamics. However, within that average there are realizations in which offloading is advantageous (high channel capacity and low processing congestion) or disadvantageous (low channel capacity and high processing congestion). In order to fully harness the performance gain edge computing can offer, while facing the dynamics of the IoT system, we develop a sequential probing and decision making framework. At each stage, the UAV observes the current realization, estimates the residual cost to complete the task, and makes a decision about whether to initiate local processing or not. This formulation corresponds to an optimal stopping time problem on a semi-Markov process.

Under the assumptions listed in the previous section, the temporal evolution of the system can be represented as a semi-Markov process. Let's define as t_j^+, $j{=}0, 1, 2, \ldots$ the time instants right after the occurrence of an event, defined as the establishment of the connection with the network, the delivery of the data to the edge server, or the completion of a computation task at the UAV or edge server. We denote the state of the system at time t_j^+ as the random variable $S(t_j^+)$. The state space $\mathcal{S}$ of $S(t_j^+)$ consists of an initial state s_0, two termination states s_{UAV} and s_{ES}, and a number of states describing data transmission and task queueing process. The termination states correspond to the computation task being completed locally at the UAV (s_{UAV}) and offloaded to the

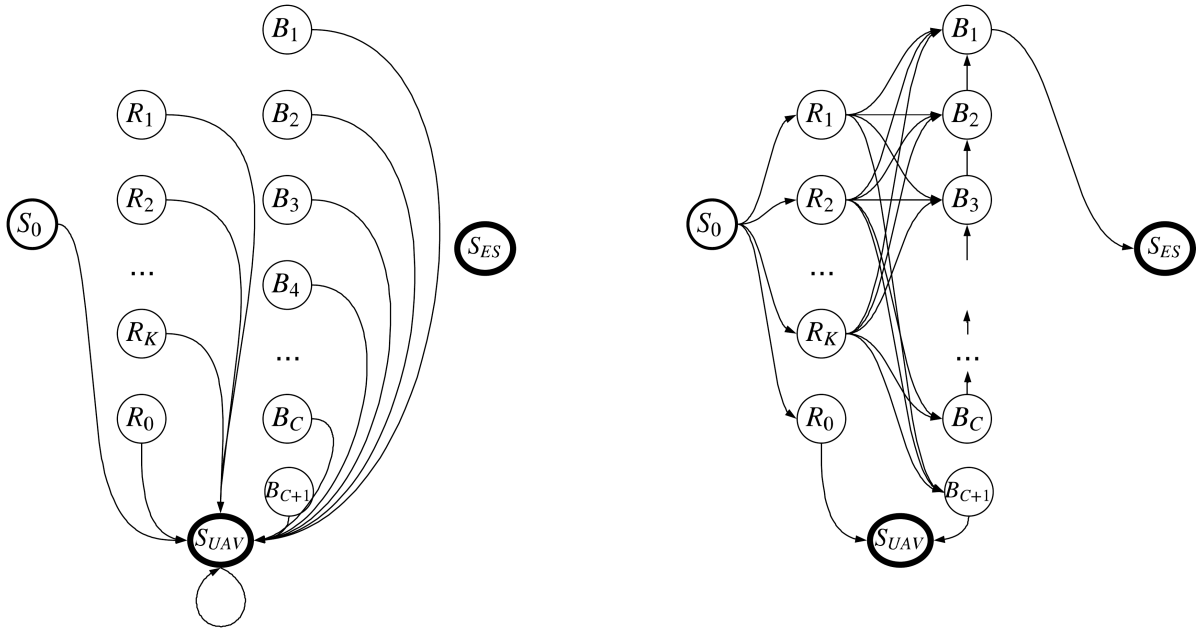


Figure 2: Representation of state transitions with non-zero probability in the Markov Chains associated with decision $u = 0$ (left) and $u = 1$ (right).

edge server ($s_{\rm ES}$). Specifically, we include *(i)* a set of $K{+}1$ states $R_0, R_1, \ldots, R_K$ associated with a transmission rate, that is, a channel state in the ranges defined in the previous section; and *(ii)* a set of $C{+}1$ states $B_1, \ldots, B_{C+1}$ associated with the position of the UAV task in the task buffer at the edge server. Note that B_{C+1} corresponds to a full buffer at arrival, that is, the UAV task is rejected. It can be shown that the process $\mathbf{S}{=}(S(t_j^+))_{j=0,1,\ldots}$ is a Markov process.

At each time instant t_j^+, the UAV is notified of the state $S(t_j^+)$ from the network access point or the edge server, and makes a binary decision $u{\in}\{0,1\}$, where 0 and 1 correspond to local computing and continuing on the edge-assisted pipeline – that is, further deferring local computing, respectively.

A. Transition Probabilities

We now describe the transition probabilities governing the dynamics of the stochastic process $\mathbf{S}$. For the sake of notation clearness, we denote the time t_j^+ with its index j. We define, then

$$P(s'|s,u){=}\Pr(S(j{+}1){=}s'|S(j){=}s, U(j){=}u). \tag{5}$$

If the decision is equal to 0, the transition probabilities from any state s are

$$P(s'|s,0){=}\begin{cases}1 & \text{if } \ s'{=}s_{\rm UAV};\\ 0 & \text{otherwise.}\end{cases} \tag{6}$$

That is, if the decision is to compute locally, the process moves to state $s_{\rm UAV}$ deterministically from any state.

We, then, analyze the transition probabilities if the decision is 1, that is, the UAV further defers the initiation of local computation. In such case, from the initial state s_0, the channel distribution is sampled, and the state moves to one of the pre-transmission states R_i with probability equal to that of the associated interval. Thus,

$$P(s'|s_0,1){=}\begin{cases}\pi_i & \text{if } \ s'{=}R_i, \ i{=}0,1,\ldots,K;\\ 0 & \text{otherwise,}\end{cases} \tag{7}$$

where $\pi(i){=}\Theta_g(g_{i+1}) - \Theta_g(g_i)$.

In any state R_i, the UAV is reported the transmission rate, that is, the index i, from the wireless access point. If the decision is to defer local processing, the transition probabilities from R_i, i=1,...,K, are

$$P(s'|R_i,1){=}\begin{cases}\sigma_{c-1} & \text{if } \ s'{=}B_c\\ 0 & \text{otherwise.}\end{cases} \tag{8}$$

σ_c is the probability that the UAV task will find c tasks stored in the edge server buffer at arrival. It is known that

$$\sigma_c{=}\frac{(1-\lambda/\mu)(\lambda/\mu)^c}{1-(\lambda/\mu)^{C+1}}. \tag{9}$$

The state R_0, corresponding to disconnection from the network, deterministically leads to $s_{\rm UAV}$.

At the beginning of any state B_c, the UAV is notified of the index c. For states B_c, $c{=}2,\ldots,C$, the transition probabilities are

$$P(s'|B_c,1){=}\begin{cases}1 & \text{if } \ s'{=}B_{c-1}\\ 0 & \text{otherwise.}\end{cases} \tag{10}$$

State B_{C+1} corresponds to a full task queue and, thus, rejection of the UAV task. Therefore, from B_{C+1} the system deterministically moves to $s_{\rm UAV}$. State B_1 corresponds to the UAV task being in the first position, and deterministically leads to $s_{\rm ES}$.

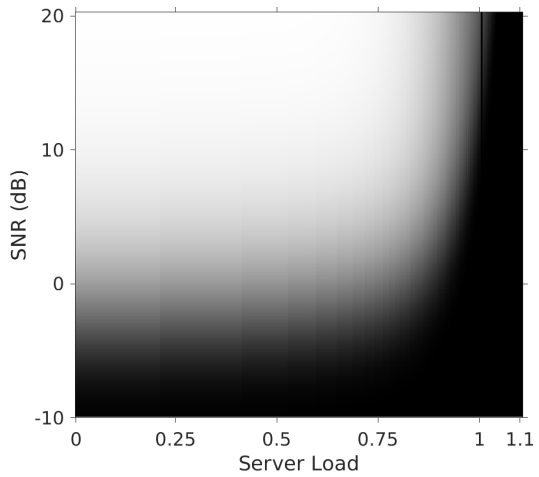


Figure 3: Probability of offloading to the edge server (lighter shades corresponds to higher probability) with $\omega = 0$.

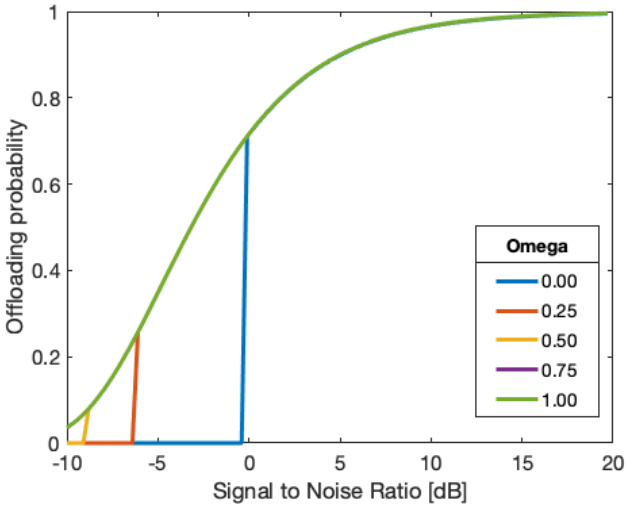


Figure 4: Probability of offloading to the edge server for different values of ω, as a function of channel quality (SNR) with $\rho = 0.5$. We use $\mu = 1/0.461/s$ to emphasize the observed effects.

B. Cost Functions and Optimal Policy

With the transition probabilities conditioned on the state and action, we can now build the optimization process. We consider a formulation where the objective of the UAV is to minimize $E(V)$, with

$$V = \omega E + (1-\omega)T, \tag{11}$$

where ω is a positive weight in $[0, 1]$.

To this aim, define the time and energy spent in state $s \in \mathcal{S}$ as $\Phi(s, u)$ and $\Psi(s, u)$ conditioned on the action u, respectively. Note that both the latter and the former are random variables. We denote their average as $\phi(s, u)=E(\Phi(s, u))$ and $\psi(s, u)=E(\Psi(s, u))$. We further define $C(s, u)=\omega\Phi(s, u) + (1-\omega)\Psi(s, u)$, with average $c(s, u)$.

The average time and energy cost associated with the initial state are equal to 0. In the termination states s_{UAV} and s_{ES}, we have

$$\phi(s_{\text{UAV}}) = 1/\mu', \tag{12}$$
$$\psi(s_{\text{UAV}}) = (P_{\text{P}} + P_{\text{H}})/\mu', \tag{13}$$

and

$$\phi(s_{\text{ES}}) = 1/\mu, \tag{14}$$
$$\psi(s_{\text{ES}}) = P_{\text{H}}/\mu. \tag{15}$$

Herein, based on actual value obtained by means of experimental evaluations, we assume that the transmission energy expense PL/R_i is negligible compared to the processing and hovering energy expense. Note that in the termination states the action is pre-determined and does not need to be formally included in the cost. From any transmission and queueing state $R_0, \ldots, R_K$ and $B_1, \ldots, B_{C+1}$, if the decision is to initiate local processing at the UAV (u=0), the process immediately moves to s_{UAV} and the energy and time cost are both equal to 0. Note that such decision is forced in states R_0 and B_{C+1}.

If the decision is to defer local processing (u=1), the costs are

$$\phi(R_i, 1) = L/R_i, \tag{16}$$
$$\psi(R_i, 1) = (P_{\text{H}} + P)L/R_i. \tag{17}$$

with i=, $1, \ldots, K$, and

$$\phi(B_i, 1) = 1/\mu, \tag{18}$$
$$\psi(B_i, 1) = P_{\text{H}}/\mu. \tag{19}$$

Herein, based on measurements obtained by means of experimental evaluations, we assume that the transmission energy PL/R_i is negligible compared to the computing and hovering energy expense.

The problem of minimizing the expected total cost can be rephrased as a Markov Decision Process over a finite temporal horizon. We aim at finding, then, the (deterministic) optimal policy $u^*(s)$, where

$$u^*(s) = \arg \min_{u=\{0,1\}} E\big(V_{\text{res}}(s, u)\big), \tag{20}$$

where $E(V_{\text{res}}(s, u))$ is the expected minimum cumulative residual cost to a termination state $s^\dagger$ from state s given that decision u is selected, that is,

$$\min_{\mathbf{U}_1^{j^\dagger}} E\left(\sum_{j=0}^{j^\dagger} c(S(j), U(j)|U(0)=u, S(0)=s)\right), \tag{21}$$

where

$$j^\dagger = \min(j : S(j) \in \{s_{\text{UAV}}, s_{\text{ES}}\}), \tag{22}$$

and $\mathbf{U}_1^{j^\dagger} = (U(0), \ldots, U(j^\dagger))$.

We compute the optimal policy using the Value Iteration method [8], which focuses on iterations producing policies achieving performance increasingly close to the optimal point. In our case, the optimal value form the starting state yields the experienced delay and energy consumption for each image, when $\omega = 0$ or $\omega = 1$ respectively. Values at other states,

at convergence, represent the expected future cost from that state obtained using the optimal policy. Let V_t be the vector whose elements are the value function for each state in the state space, where t is the number of steps taken in the recursion. Then $V_t \in \mathcal{R}^{5+K+C}$, where the size of the vector derives directly from the states definition in Figure 2. The arbitrarily initialized vector V_0, is then updated using the Bellman equation as follows:

$$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s')$$
$$V_k(s) = \min_a Q_k(s,a), \tag{23}$$

where $Q_t(s,a)$ represents the expected cost taking action a. Note that, to reduce the time to convergence, the updates are usually computed backwards, from the end nodes (in our case $S_{ES},\ S_{UAV}$) to the input node S_0.

V. Numerical Results

This section presents and discusses results obtained using the model and optimization technique proposed in this paper. First, we analyze performance metrics and offloading probabilities exploring parameters describing channel quality and server load. We, then, characterize the performance of the proposed scheme in a realistic environment, where channel parameters (that is, SNR) is obtained based on trajectory of the UAV. Note that in this latter section of the results, we can analyze the temporal behavior of the system during the mission.

To make our observations more meaningful, we derive the optimal policies under different channel and load conditions using parameters obtained from real-word experimentation. These values are used unless otherwise stated. Specifically, we used a 3DR Solo Drone mounting a Pixhawk flight controller running ArduCopter connected to a Raspberry Pi model 3B as companion computer. We use as edge server a Laptop with 16GB RAM and Intel Core i7-6700HQ processor with Nvidia GM204M GPU. We set the number of pictures collected in each location to 1, where each picture has resolution equal to 720×480. The average size of each picture after encoding is 80 KB. The pictures are processed implementing a face recognition algorithm using a multi-scale Haar Cascade, which takes on average $1/\mu'$=0.56s at the UAV and $1/\mu$=0.046s at the edge server. We consider SNR values in the range $[-10, 20]$ dB and transmission rates in the range from 1 Mbps to 11 Mbps (matching a system using Wi-Fi IEEE 802.11). Power consumption rates are based on battery level readings in the same set up: in particular, we set $P_h = 0.1$ levels/s, $P_p = 10\% \cdot P_h$ levels/s. The optimal deterministic policy $\mathbf{U}_1^{j^\dagger}$ given the parameters is computed using Equation (21). We note here that our performance analysis will be impacted by an error due to the resolution chosen on the set of available rates and positions. As a result of the position error, a different expected SNR will be used, causing a difference in the probability distribution over the rates. These will be averaged out on multiple runs and so our results still hold. We consider on the other hand the rate resolution error in this case acceptable: the difference in Mbps in the Wifi protocol is much higher than the one considered. Our investigation takes into consideration several aspects of the system, and even though the error margin at small rates might is large ($100ms$), it is worth noting that in all other scenarios it reduces to a few milliseconds. Furthermore the overall policy behaviour does not change due to these effects, and in this respect while absolute values might differ, the statistics and system behaviour will be consistent with our results.

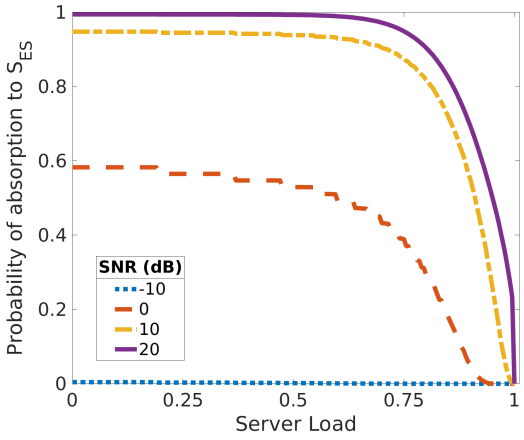


Figure 5: Probability of offloading the computation to the edge server as a function of the server load ρ.

A. Performance Analysis

In Figure 3, we show the probability of offloading to the Edge Server as a function of SNR and server load $\rho = \lambda/\mu$. This probability corresponds to the probability of the process being absorbed in S_{ES} from S_0 conditioned on the control policy, defined as

$$P^{\infty}_{S_0}(Y) = \lim_{t\to\infty} P(S(t) = S_Y | S(0) = S_0, U = \mathbf{U}_1^{j^\dagger}) \tag{24}$$

where $Y \in \{S_{\text{UAV}}, S_{\text{ES}}\}$, and $P^{\infty}_{S_0}(S_{\text{UAV}}) + P^{\infty}_{S_0}(S_{\text{ES}})$=1. In Figure 3, we plot $P^{\infty}_{S_0}(S_{ES})$, using lighter pixel color for higher probabilities. As expected, for low values of ρ and high values of the SNR, the offloading probability is almost equal to 1, that is, the UAV offloads computation when system conditions are favorable. When the SNR is sufficiently low, the UAV will likely be disconnected, or the cost of offloading might exceed that of local computation due to the large time needed to transport the data to the edge server. Similarly, if the load parameter ρ is large, that is, the ES buffer has frequent arrivals or computation tasks take a large time to be completed, the UAV chooses to compute locally.

In Figure 4 we show the effect of ω, the parameter that controls the tradeoff between energy and delay in the objective function, over the optimal policy and consequently on the offloading probability. In the plot, each line corresponds to a different value of $\omega \in [0, 1]$, where the larger ω the larger the weight of energy cost. The impact of including energy in the optimization is apparent: the larger ω, the larger the offloading probability, even for low SNR, where transmitting over the channel may result in an increased overall delay. In

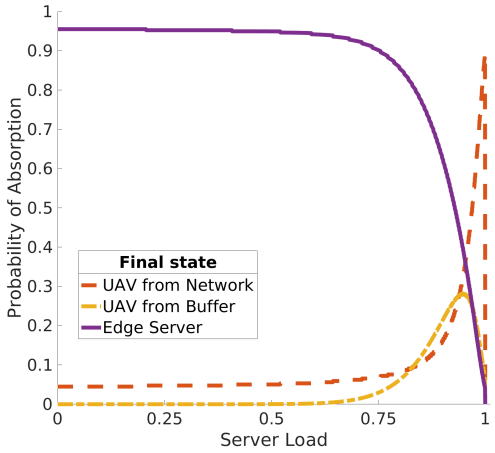
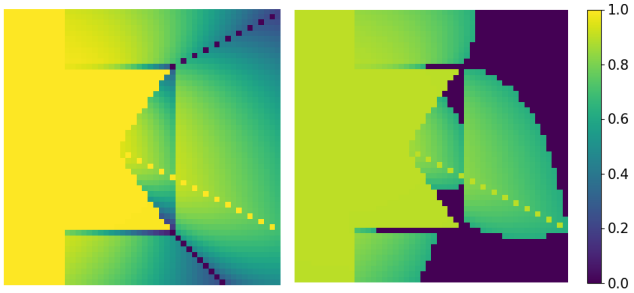


Figure 6: Probability of selecting local computation in the three main decision stages or offloading to the edge server.



(a) (b)

Figure 7: Offloading probability on the considered map. The average SNR is set to 9dB and server load set to (a) 0% and (b) 70%.

fact, while a larger delay leads to a larger hovering time, and thus more hovering energy expense, offloading eliminates the energy cost associated with local processing. We observe an interesting threshold effect, where the policy transitions from fully local computing to partial offloading at an SNR value which is a function of ω.

In Figure 5 we fix the SNR, and show $P^{\infty}_{S_0}(S_{ES})$ as a function of ρ. As the SNR decreases, the probability of offloading to the edge server decreases as well. Intuitively, the SNR influences the shape of the probability curve. Interestingly, high SNR values show a sharp transition from offloading to local computing, whereas low SNR values have a more progressive transition, most likely due to the distribution of the communication time.

Finally we illustrate the value of probing compared to a simple decision informed by the average delay pre-computed based on a priori knowledge of the parameters. Figure 6 shows the probability that the decision of processing locally is taken at the different stages or that offloading is selected. Specifically, the decision stages are:

- **Stage 0:** the initial stage S_0, where the UAV knows the parameters, but not the channel or queue state;
- **Stage 1:** R_i, where the UAV has connected with the network and is aware of the maximum transmission rate;

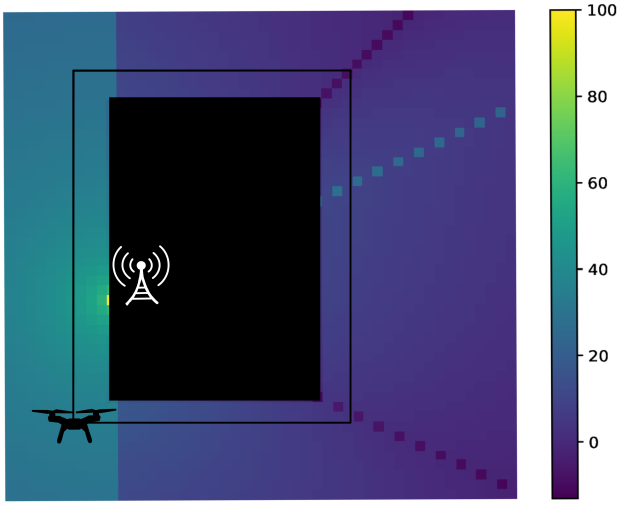


Figure 8: Map of the considered area, centered around a 30 m high building. Symbols display the access point's placement and the drone trajectory. Different shades show Signal To Noise ratio in dB across the area.

- **Stage 2:** B_j, where the UAV reached the edge server, that is, upon transmission after transmission, and is reported the position in the processing queue.

For small values of ρ, offloading is predominant, with a small probability of local computing decision forced by extremely poor channel conditions. As ρ increases, the set of rates corresponding to local computing decisions increases. In fact, the delay requirement for the data transportation becomes more stringent as the average time spent in the edge server buffer increases. In the transition phase between offloading and local computation, we can observe a spike in the probability that the UAV will select local computing after the edge server is reached, as the probability finding a number of tasks in the buffer sufficiently large to make offloading disadvantageous increases before a region in which probing is not even attempted. The policies resulting from the abstraction of the system we adopt have a simple structure. Across the phases of the decision making, the agent will identify thresholds within layers of the Markov process corresponding to binary decisions. While the structure is simple, the thresholds are function of the distributions of channel quality and incoming load at the server.

B. Mission Trajectory

We now analyze a mission trajectory of the UAV in an urban scenario inspired by applications such as city monitoring and building inspection. We consider the trajectory illustrated in Fig. 8, where the UAV flies at fixed altitude and constant distance from the building's external surface in a loop starting from the lower left corner and proceeding in clockwise direction. The map shown has delimits a 50×50 meters area, where both the access point and the UAV are at $15m$ altitude, and the building's width, length and height are 20m, 30m, 30m.

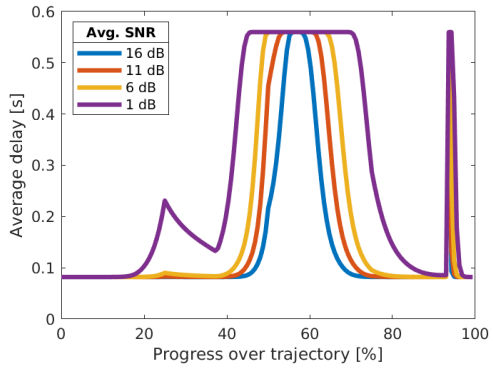


Figure 9: End-to-end delay over the described trajectory for different values of external interference with no server load.

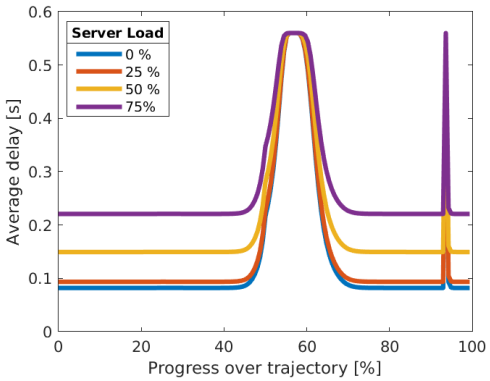


Figure 10: End-to-end delay over the trajectory for different values of server load with average SNR of 16dB.

In the scenario, we consider a setting with one access point and one building, and we compute the SNR – to be plugged in our model, see Eq. (3) – using the building shadowing model [9] (added to the attenuation caused by free-space propagation)

$$L = \alpha n + \beta d_0, \tag{25}$$

where α is the attenuation per wall (dB), n is the number of walls penetrated, β is the attenuation per meter (dB) and d_o the distance in meters traveled through obstacles. We use experimentally validated coefficients in [9] $\alpha = 9dBm$ and $\beta = 0.9dB/m$

Using the same set of parameters as in the previous set of results, we find the optimal policy for each position in the considered map. In Fig. 7, we show how the probability of offloading to the edge server evolves along the trajectory. In Fig. 7.a, the edge server is dedicated to the UAV. The impact of the additional attenuation effect of the building on the strategy is apparent: the offloading probability decreases in regions that are more affected by the additional attenuation. Overall, the low server load leads to offloading being a predominant strategy. Fig. 7.b shows the same map where we increase the server load to 70%, and we can see that the adaptive scheme reacts selecting edge computing as the best strategy in a smaller fraction of realizations. This effect is due to the higher chances that the task generated by the UAV will find several other tasks in the server's buffer.

We now consider the delay performance over the full trajectory, and illustrate how the strategy evolves. In Fig. 9, we plot the average capture-to-output delay achieved by the optimal strategy. We can observe how, for different values of SNR, low performance regions expand. This is due to the higher probability that local computing will be chosen due to the low data rate supported by the channel. Interestingly, we can observe how new spikes and low performance regions emerge as noise increases and the strategy switches to different modalities in some regions.

The server load ρ has a much different impact, as shown in Fig. 10. The average delay increases homogeneously along the trajectory to reach a cap determined by the policy always choosing local analysis. Additionally, we observe how moderate server loads have relatively low impact (*e.g.*, 25% vs 50%).

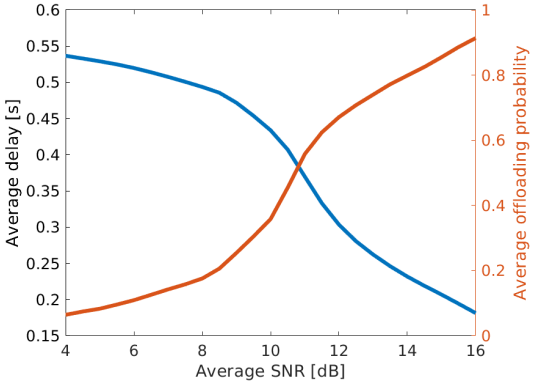


Figure 11: Delay average over the full trajectory for varying average SNR for both only offloading policy and our approach. We also plot the probability for the UAV to successfully offload in our schema.

Considering the average over the full trajectory, we now explore how some parameters affect performance and strategies. In Fig. 11 and Fig. 12, we display the average delay (in blue) and offloading probability (in orange), averaged over the trajectory, for different noise levels and edge server's load. Interestingly, while the delay has a clear inverse relationship with the offloading probability when varying average SNR, the relationship between average delay and offloading probability is less marked when varying the edge server's load. In the former plot (Fig. 11), we have a sharp change at $\approx 10dB$, where the probability of a successful offload sharply decreases and the delay increases due to the more frequent selection of local processing. In the latter plot (12), we observe a low sensitivity of the delay, where in the low to moderate load region we have an increasing average delay, but a minor change in the offloading strategy. The delay experiences a sharp increase when the offloading probability sharply decreases in the high load region. This effect is due to the more gradual degradation imposed by increasing load compared to that of a worsening SNR.

We now characterize the impact of the computing capacity of the devices, expressed as their service rates. We explore a

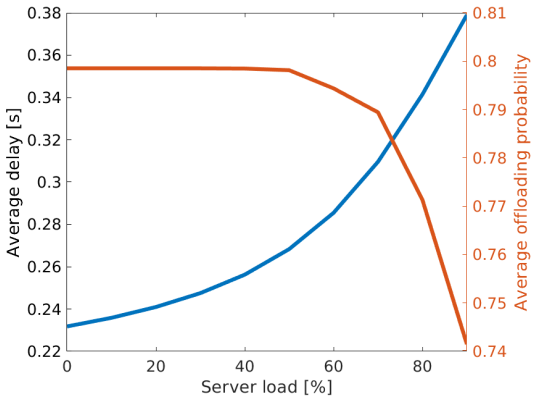


Figure 12: Delay average over the full trajectory for varying server load, and the probability to successfully offload.

range of values of $\mu \in [1, 40]$ for the edge server, $\mu' \in [1, 8]$ for the UAV, to evaluate the impact of different design choices and operational settings. We highlight the advantage of an adaptive approach, by depicting the percent delay increase when a fixed offloading policy is used. We remark that we are still considering averages over the entire trajectory.

The gain is shown in Fig. 14, where we we set the load to 0 (a) and 70% (b). Note in both graphs that higher gains (darker regions), are focused in high local service rates and relatively low edge server service rates areas. This conveys the fact that adaptation can improve performance only when the strategy is non-trivial and adaptation can bring benefit. When local service rate is small (left portion of the graphs), we indeed observe than the offloading policy,

Interestingly, we see how the gain granted by the adaptive strategy is higher when the load is higher, this due to the fact that our policy can effectively fall back to local computing when the buffer is busy in specific realizations of the process.

In Fig 15.a we can see how the average delay using an adaptive policy is very sensitive to the local processing capacity, but has a weak dependency on the edge server processing capacity. However, as shown in Fig 15.b, we can see that the delay's variance is higher in the areas where the gain is small. In fact, the advantage of the adaptive technique is to choose local processing whenever it seems advantageous, and that is shown in the areas where lower variability maps to local processing being the optimal policy.

C. Characterization of State in Temporally Correlated Environment

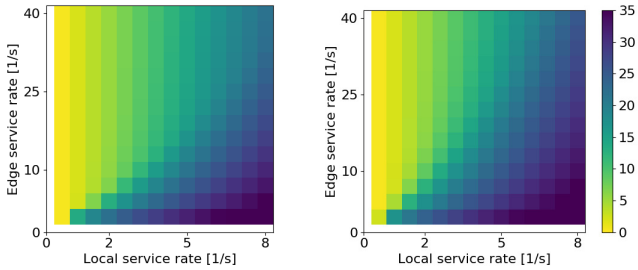
We built on these results and created an event based simulator that allows us to capture the temporal correlations between subsequent positions along the trajectory. We use this tool in order to study the impact of different state representations on the performance of a decision agent.

In the state representations we include:

- phase: before offloading, at the edge server or processing locally
- position: the (x, y) coordinates on the map
- E[SNR]: average SNR at the current position

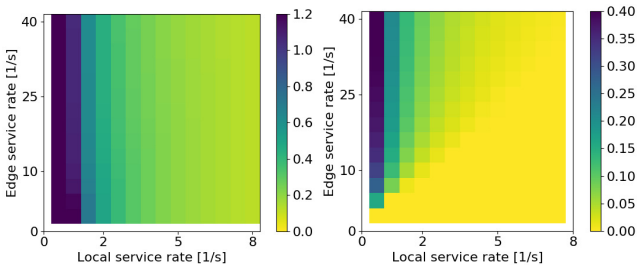
(a) average SNR of 16dB (b) average SNR of 6dB

Figure 13: Delay improvement using adaptive schema compared to constant offloading at different average SNRs.



(a) $\rho = 0\%$ (b) $\rho = 70\%$

Figure 14: Gain percentage over a full trajectory for different hardware configurations. Processing speeds are referenced as serving rates μ. Both cases have average SNR at 16 dB, but they differ in edge load ρ.



(a) (b)

Figure 15: (a) Delay and (b) standard deviation for delay over a trajectory for different hardware configurations with average SNR of $16dB$ and no server load. Processing speeds are referenced as serving rates μ.

- num. jobs: number of jobs in the queue
- past (action, delay) tuples: we include the last 3 actions and delays.

Although in a collaborative system we would expect to collect all the above, if using a third-party infrastructure or different networking protocols, it could be difficult or impossible to collect some of this dynamic information. For this reason we study the same system using three different observed states:

1) full state
2) phase, position, past action-delays
3) phase, past action-delays

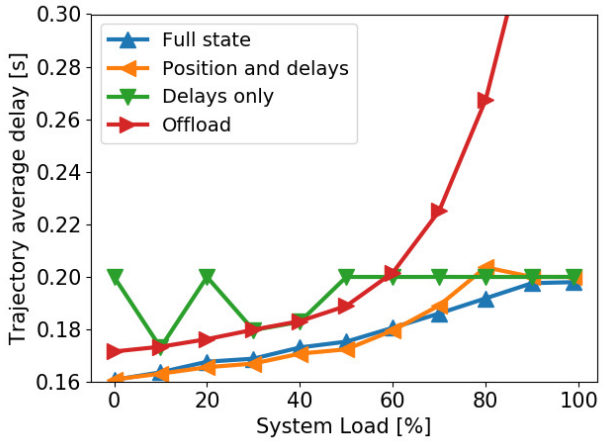


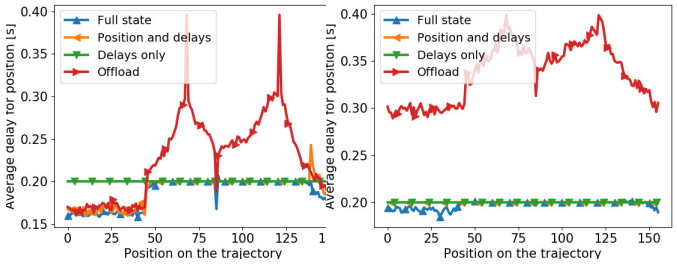
Figure 16: Average delay over the full trajectory for different server loads shows the adaptability of agents that have spatial awareness.

Due to the nature of this state, where some elements are discrete, other continuous, we involve the use of a function approximator, such as Deep Neural Networks (DNN), to learn the q-function. In order to learn from experience in this new setting, we resort to some techniques investigated in Deep Reinforcement Learning (DRL). DRL has been successful in environments such as Atari games [10], the game of Go [11] and many others [12]. As shown in these works, there are many details that help DRL converging to a stable approximation for the q-function. In fact naive approaches often do not work, due to the inherent difference in the way DNNs learn with respect to the tabular approach. For example, DNNs are subject to catastrophic forgetting [13], the tendency of forgetting early examples seen in the training procedure. To address this problem Schaul et al. introduced replay buffer [14], a buffer where samples are stored after observation. We use samples randomly extracted from the buffer, and feed them to the DNN in batches. The model will compute the function $Q(\cdot)$, and return a vector with q-values for each of the actions. However since we can take only one action, we will observe only one reward, and therefore we will update, i.e. compute the loss and apply back-propagation, only relative to one output. We use the Bellman equation based on one step Temporal Difference to compute the estimate q-value for the next step:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \operatorname*{argmax}_{a'} \hat{Q}(s',a')) \tag{26}$$

where α is learning rate, γ is discounting factor, and $\hat{Q}$ is an old version of the DNN. We use to different networks $\hat{Q}, Q$, so that the q-value estimation does not vary too quickly, causing divergent behaviour. We update $\hat{Q} = Q$ periodically to improve our approximation of the q-function.

As mentioned, we maintain the setup the same as in the previous setting, where the agent has to decide whether to continue offloading or not at two stages: when it arrives to a position and when it gets in queue to the edge server.



(a) $\rho = 70\%$ (b) $\rho = 90\%$

Figure 17: Delay over the trajectory for different state representations at different server loads.

From the agent's point of view all transitions are probabilistic, since the rate might be 0 and the queue might be full. In order to discourage fixed policy of only offloading, we have a small network probing term that is added to the delay when a transmission is unsuccessful.

We trained different agents using the state representation described earlier, and obtained insights into what carries useful information in predicting the optimal policy at each position. In Fig. 16 we display the average delay of each of the agents going through the trajectory. We can observe that using all the information available gives the agent an advantage similar to what we observed earlier, since it is able to choose the correct policy for each given channel situation. Interestingly, removing average SNR and position in queue does not influence the performance for loads lesser than 70%. There we see the two lines (blue for full state and orange for only position and delays) diverging, with the less informative state having higher average delay. The critical advantage of these two policies on delays only and fixed policies, can be observed in Fig. 17: we notice that the central part of the trajectory, where the line of sight is obstructed by the building, offload strategy incur in very high delays. For higher loads, we can also see how the position in queue at the edge server changes the ability of the user to exploit offloading only when advantageous: in Fig. 17 we see that only the agent that has access to the position in the server buffer is able to exploit successfully offloading in positions 0-50 in high load regimes.

This study reveals how the position along the trajectory is a viable proxy for the average Signal to Noise Ratio, and how on the other hand previous delays are not as predictive of the number of processes in queue at the server. Furthermore this opens the opportunity for further investigation in spatial-temporal maps that can embed this information and allow dynamic adaptation in continuous learning settings.

VI. Related work

In the recent years, there have been many contributions addressing task offloading for UAVs. The proposed framework builds on the many contributions studying offloading for mobile devices. However, as mentioned in [15] and other works, mobile applications usually assume a sparse and inhomogeneous generation of computing tasks. In mission-oriented and robotic applications, it is usually assumed that

the application generates a continuous flow of homogeneous computing tasks. Furthermore, as seen in [16], deadlines in UAV systems, differently from mobile applications, are hard deadlines. The framework we propose embeds these features at its core to improve latency. The sequential decision making controls latency at a fine grain ti meet stringent constraints. The RL framework we propose takes advantage of the continuous flow of tasks to learn space-time properties of optimal offloading and improve prediction using spatio-temporal correlation in the system. Many contributions in this area, such as [17] and [18], focus on the problem of finding the best edge server defined over long-term performance energy, delay and throughput metrics.

Other contributions, such as, [19], focus, instead, on monetary metrics (for instance associated with communications) to guide the optimization process. In [19] a game theoretic approach is proposed, with which the authors are able to reduce the communication cost, considering not only multiple servers, but also the possibility for some of them to offload the computation further. Our work centers on short-term metrics, which allow a fine degree of control when optimizing the offloading process. We remark how this is a marked difference with respect to most existing literature.

Very recent contributions, such as [20], focus on the optimization of specific scenarios, assuming the a priori knowledge of a prediction model for the channel state, which leads to sophisticated decision making algorithms to determine which part of the task can be offloaded. This interesting class of approaches imposes stronger limitations on the type of analysis task. [21] presents an approach based on fuzzy logic to face the high uncertainty induced by these applications.

In [22], B.Liu et al. propose to offload computationally intensive tasks from UAVs to edge and cloud servers. They formulate a joint computation and routing optimization by defining a three-layered computational model on which they design a polynomial near-optimal approximation algorithm. The authors use a Markov approximation technique described in [23], which is useful when solving network combinatorial optimization problems. However, they do not consider energy-related metrics and control.

Energy expense is considered in the work by Zhu et al. [24], where cooperative approach to computation offloading for UAVs is presented that aims to improve the inefficiency of naive local computing solutions. The authors explore an urban environment for UAV operations, and use simulated annealing to minimize the energy consumption while satisfying a delay constraint. Our solution jointly optimizes energy and delay, offering more flexibility to the application.

Our work can also be connected to practical contributions, where, instead of rigorous optimal decision making, different heuristics are used to control task offloading. In [7], a parametric threshold based algorithm is used to decide not only if offloading can be beneficial, but also matching with the right server and adaptively probe available remote computing options.

VII. Conclusions

In this paper, we presented a framework to control processing task offloading to edge servers in UAV systems. The framework sequentially probes the state of the network and of the edge server buffer to make optimal decisions between local and edge-assisted computing. Numerical results show the delay reduction granted by the proposed technique, which is based on a Markov Decision Process formulation solving an optimal stopping time problem, compared to non-adaptive strategies. Finally we validate our results on Network Simulator 3 in a realistic task such as building inspection.

References

[1] K. Ro, J.-S. Oh, and L. Dong, "Lessons learned: Application of small uav for urban highway traffic monitoring," in *45th AIAA aerospace sciences meeting and exhibit*, 2007, pp. 2007–596.

[2] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, "A UAV system for inspection of industrial facilities," in *IEEE Aerospace Conference*, 2013, pp. 1–8.

[3] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision agriculture*, vol. 13, no. 6, pp. 693–712, 2012.

[4] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging uavs for disaster management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[5] S. ur Rahman, G.-H. Kim, Y.-Z. Cho, and A. Khan, "Deployment of an SDN-based UAV network: Controller placement and tradeoff between control overhead and delay," in *International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2017, pp. 1290–1292.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[7] D. Callegaro, S. Baidya, and M. Levorato, "Dynamic distributed computing for Infrastructure-Assisted autonomous uavs," in *2020 IEEE International Conference on Communications (ICC): SAC Tactile Internet Track (IEEE ICC'20 - SAC-10 TI Track)*, Dublin, Ireland, Jun. 2020.

[8] R. Bellman, "Dynamic programming and lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

[9] S. E. Carpenter and M. L. Sichitiu, "An obstacle model implementation for evaluating radio shadowing with ns-3," in *Proceedings of the 2015 Workshop on Ns-3*, ser. WNS3 15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1724. [Online]. Available: https://doi.org/10.1145/2756509.2756512

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv*, 2013, cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[12] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv preprint arXiv:1509.06461*, 2015.

[13] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[14] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[15] 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, 2019.

[16] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "Uav trajectory planning for data collection from time-constrained iot devices," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 34–46, 2020.

[17] K. Kim and C. S. Hong, "Optimal task-uav-edge matching for computation offloading in uav assisted mobile edge computing," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019, pp. 1–4.
[18] K. Kim, Y. M. Park, and C. Seon Hong, "Machine learning based edge-assisted uav computation offloading for data analyzing," in *2020 International Conference on Information Networking (ICOIN)*, 2020, pp. 117–120.
[19] A. Alioua, H. eddine Djeghri, M. E. T. Cherif, S.-M. Senouci, and H. Sedjelmaci, "Uavs for traffic monitoring: A sequential game-based computation offloading/sharing approach," *Computer Networks*, vol. 177, p. 107273, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619315798
[20] J. Chen, S. Chen, S. Luo, Q. Wang, B. Cao, and X. Li, "An intelligent task offloading algorithm (itoa) for uav edge computing network," *Digital Communications and Networks*, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864819303037
[21] A. Karanika, P. Oikonomou, K. Kolomvatsos, and T. Loukopoulos, "A demand-driven, proactive tasks management model at the edge," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2020, pp. 1–8.
[22] B. Liu, H. Huang, S. Guo, W. Chen, and Z. Zheng, "Joint computation offloading and routing optimization for uav-edge-cloud computing environments," in *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 1745–1752.
[23] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.
[24] S. Zhu, L. Gui, J. Chen, Q. Zhang, and N. Zhang, "Cooperative computation offloading for uavs: A joint radio and computing resource allocation approach," in *2018 IEEE International Conference on Edge Computing (EDGE)*, 2018, pp. 74–79.

Davide Callegaro obtained both his B.S. (Information Engineering) and M.S. (Computer Engineering) from University of Padova, Italy, in 2013 and 2016, respectively. In 2015 he visited Polytechnic University of Catalonia for a semester as part of his Master. In Fall 2016 he started his Ph.D in Computer Science, where he currently works as a Ph.D Candidate, under the supervision of Prof. Marco Levorato. He is a member of IEEE and IEEE ComSoc society. He is interested in optimization and control in real world embedded systems. He interned in Bosch and Nutanix as a Research Intern, investigating how to optimize performance and reliability of existing platforms. In his doctorate he is focusing on task allocation in real-time heterogeneous distributed systems, investigating trade-offs in robotic systems relying on edge computing infrastructure.

Marco Levorato joined the Computer Science department at University of California, Irvine in August 2013. Between 2010 and 2012, He was a post-doctoral researcher with a joint affiliation at Stanford and the University of Southern California. From January to August 2013, he was an Access post-doctoral affiliate at the Access center, Royal Institute of Technology, Stockholm. He is a member of the ACM, IEEE and IEEE Comsoc society. His research interests are focused on next-generation wireless networks, autonomous systems, Internet of Things, and e-health. He has co-authored over 100 technical articles on these topics, including the paper that has received the best paper award at IEEE GLOBECOM (2012). He completed the PhD in Electrical Engineering at the University of Padova, Italy, in 2009. He obtained the B.S. and M.S. in Electrical Engineering summa cum laude at the University of Ferrara, Italy in 2005 and 2003, respectively. In 2016, he received the UC Hellman Foundation Award for his research on Smart City IoT infrastructures and the Dean Research award in 2019.