

Optimal Task Offloading Policy in Edge Computing Systems with Firm Deadlines

Khai Doan*, Wesley Araujo*, Evangelos Kranakis[†], Ioannis Lambadaris*, Yannis Viniotis[‡]

**Carleton University, Department of Systems and Computer Engineering, Ottawa, ON, Canada.*

[†]Carleton University, School of Computer Science, Ottawa, ON, Canada.

[‡]North Carolina State University, Department of Electrical and Computer Engineering, Raleigh, NC, USA.

Email: {khaidoan@sce, wesleyaraujo@cmail, kranakis@scs, ioannis@sce}.carleton.ca, candice@ncsu.edu

Abstract

The recent drastic increase in mobile data traffic has pushed the mobile edge computing systems to the limit of their capacity. A promising solution is the task migration to remote servers. Key factors to be considered in the design of offloading schemes must include the number of tasks waiting in the system as well as their corresponding deadlines. An appropriate system cost which is used as an objective function to be minimized comprises two parts. First, an offloading cost which can be interpreted as the cost of using computational resources at the external server. Second, a penalty cost due to potential task expiration. In order to minimize the expected (time average) cost over a time horizon, we formulate a Dynamic Programming (DP) equation and analyze it to describe properties of a candidate optimal offloading policy. The DP equation suffers from the well-known “Curse of Dimensionality” that makes computations intractable, especially when the state space is infinite. In order to reduce the computational burden, we identify three important properties of the optimal policy. Based on these properties, we show that it suffices to evaluate the DP equation on a finite subset of the state space only. We then show that the optimal task offloading decision associated with a state can be inferred from the decision taken at its “adjacent” states, further reducing the computational load. Finally, we provide numerical results to evaluate the influence of different parameters on the system performance as well as verify the theoretical results.

I. INTRODUCTION

Mobile-Edge Computing (MEC) and Mobile Cloud Computing (MCC) are important paradigms in addressing the limited computational capability of mobile devices. In MEC, a

remote server is placed physically near the device for computational tasks to be offloaded to the MEC-server for remote computation. MEC is similar to MCC, however the server in the latter case, is not necessarily physically close to the device. MEC is more appropriate if the network latency or network congestion is a problem. In general, there are two scenarios when MEC is most appropriate: when the application has real-time constraints or when the user/wireless device has limited resources such as memory, storage, CPU, etc. Surveys on MEC and MCC can be found in [1] and in [2], respectively.

A. Motivation and Related Works

Exploiting the advantage of offloading systems, computational services requested by users can be either processed at local servers (e.g., MEC) or offloaded onto remote servers (e.g., MCC) that have more computational capability. This feature not only improves users' quality of experience by reducing the processing, latency and power consumption, but also allows different types of applications and services to be deployed on devices with low computational capability. Due to significant advances in practical applications, analyzing MEC and MCC systems has been attracting a lot of attention in the research literature. For example, [3] proposed an optimal task offloading algorithm by maximizing an appropriate utility function. Such a function increases with users' satisfaction characterized by tasks processed remotely, and decreases with respect to the total amount of computation and energy consumption overhead. In [4], the authors study the computational complexity of task offloading policies in a MCC context where tasks have hard deadlines. In [5], they study the problem of computational offloading in a MCC context with the goal of minimizing energy consumption of the user device by taking into account its multi-core architecture.

In practical systems, the task offloading feature may suffer from uncertainty due to certain factors such as network congestion and limited computational resources of remote servers. Such factors may impose randomness in the availability of task migration. Systems that have firm task deadlines require more attention when task migration is not always available. These issues have not been adequately addressed in existing works which typically assume that task offloading can always be controlled. In [6], the authors study the problem of offloading tasks to a cloud computing infrastructure aiming at minimizing the energy consumption of the offloading device and meeting the task deadlines. In [7], the authors consider computational offloading in a real-time MEC setting in which tasks have hard deadlines. In [8], the authors

study the problem of computational offloading in the context of a mixed MEC and MCC system with the goal of minimizing the weighted cost of task processing delay and energy consumption of the devices.

Furthermore, despite several offloading algorithms proposing improvements for the system performance under different contexts, to the best of our knowledge, there is a lack of studies for the characterization of the optimal policy for task offloading to enhance the system performance. In [9], [10], the authors consider the problem of a user offloading tasks to multiple moving edge devices so as to maximize the utility of the task execution and minimize the energy consumption. In the first paper, a Deep Reinforcement Learning based method is used to solve the problem, while in the second paper, Q-learning is used.

B. Model Novelty and Main Contributions

What differentiates our model from previous studies is the combination of: (a) the randomness in the connection between both the remote server and local device, (b) tasks that have firm deadlines, and, (c) the formulation of the problem in the context of an optimal stochastic control (DP) framework. As is well-known in DP formulations, computing the solution of the DP equation may become computationally prohibitive (also known as the “dimensionality curse” of DP). With this motivation, our contributions can be summarized as follows.

- Our main contribution is the mathematical characterization of the structure of the optimal policy. The findings are presented in Theorems 1 and 2. The properties we prove in these theorems help reduce the computational burden associated with the DP equation.
- Our second contribution is a further reduction in the computational effort for determining the minimum expected (time average) cost. In Proposition 1 we derive a mathematical property of this cost. This will enable us to compute it by applying the computationally heavy DP equation to *only a finite number* of states (the so-called *lean states*); the cost for an arbitrary state can then be calculated via a much simpler algebraic computation.

The rest of this article is organized as follows. In the next section, we describe the system model under consideration. In Section III, we formulate the DP equation to minimize the expected (time average) cost. In Section IV, we introduce the notions of *reduced* and *lean*

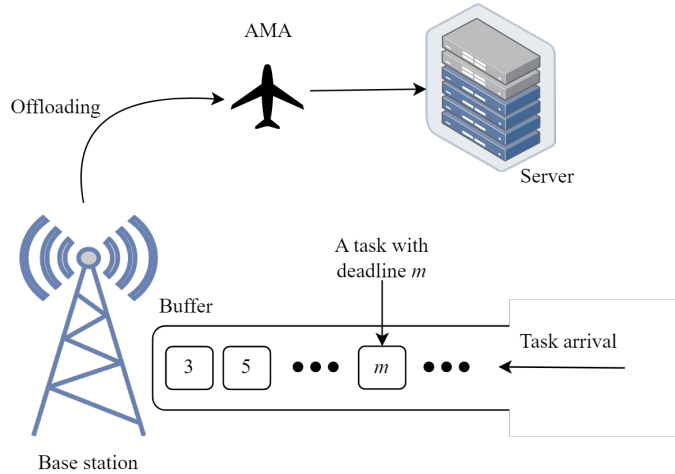


Fig. 1: System model.

state space and show how they are used to reduce the computational load for the calculation of the optimal cost of our model. Next, we study properties of the optimal policy and the minimum expected time-average (ETA) cost in Section V. The optimal policy is explicitly presented in Section VI. Subsequently, we illustrate numerical results in Section VII. We conclude our work with Section VIII. Finally, we provide proofs for all the theorems, lemmas, and propositions in Appendices presented in Sections IX, X, and XI, respectively.

II. SYSTEM MODEL

The system we consider is depicted in Fig. 1; it consists of a base station (BS) operating in discrete-time over T time slots. Tasks sent by users in the area will be buffered at the BS. We assume that the maximum deadline for any task is N , a fixed positive integer. The deadline represents the number of time slots (including the current one) within which the tasks need to be processed, otherwise they will expire and result in a penalty C_p per expired task.

We define the system state vector at a given time slot t by

$$\mathbf{s}_t = (n_1^{(t)}, n_2^{(t)}, \dots, n_N^{(t)}), t \in \{0, 1, \dots, T-1\} \quad (1)$$

where $n_i^{(t)} \in \mathbb{Z}_+$ represents the number of tasks having deadline $i = 1, 2, \dots, N$, buffered at the BS at time slot t . The state space for this system is \mathbb{Z}_+^N , which is an N -dimensional infinite state space. In this system, a number of different events takes place and triggers the

system to transit to a different state in the next time slot. Those events can be listed in order as follows.

- 1) **Connection to a remote server:** A task offloading service is provided by an external server that is outside the communication distance of the BS. The server is accessible by means of an *autonomous mobile agent* (AMA). The AMA randomly arrives in the area and behaves as an intermediate node to connect the BS and the server. With the AMA, the BS can offload tasks to the server at cost $C_o > 0$ per task. Let p_u be the probability that the AMA arrives in each time slot. The remote server is assumed to be capable of processing all tasks instantly without delay. In addition, we also assume that the task transferring from the BS to the AMA, and then, to the remote server incurs no delay.
- 2) **The deadline shifting:** The deadlines of all tasks are reduced by one when transitioning from a time slot to the next. The deadlines are strict: a penalty of C_p is incurred for every task whose deadline has dropped to 0. In this model, we assume that $C_p > C_o$.
- 3) **The arrival of a new task:** During our system operation, there will be new tasks sent by users; we assume at most one new task can arrive per time slot. We use p_i to denote the probability that a new task will have deadline i ; p_0 denotes the probability of no task arrival. Thus, the task arrival probability vector is given by $\mathbf{p} = (p_0, p_1, p_2, \dots, p_N)$. We will assume that task arrivals in different time slots are independent.
- 4) **The local processing service:** We assume that the BS has limited computational capability. Therefore, the local processing service provided by the BS is assumed to be available at random with probability μ in each time slot. When it is available, at most one task can be processed per slot.

The sequence of the above events for our model is illustrated in Fig. 2. The randomness in local processing and in the appearance of the AMA accounts for the uncertainty in the task processing.

The system cost is described by the offloading and task expiration penalty costs. To be more specific, assume that state \mathbf{s}_t (as defined in Eq. (1)) is encountered at time slot t . By L_t we denote the number of tasks that are offloaded when the system is in state \mathbf{s}_t . L_t is also called the *offloading decision*; determining its optimal value (to be defined shortly) is

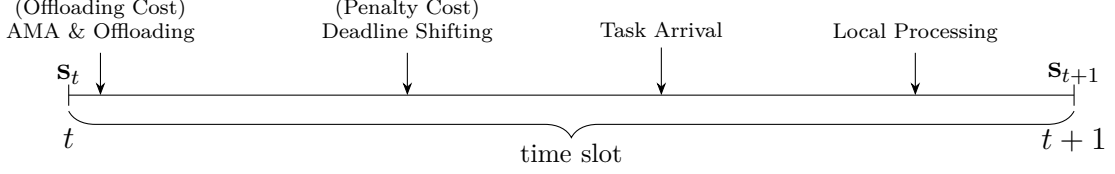


Fig. 2: Events taking place in a time slot.

the subject of this work. L_t can take values in the set

$$\mathbb{L}(\mathbf{s}_t) = \left\{ 0, 1, 2, \dots, \sum_{i=1}^N n_i^{(t)} \right\}. \quad (2)$$

In our model, all tasks share the same offloading cost C_o , and result in the same penalty C_p when expire. Trivially, it is optimal to locally process and offload tasks with the most imminent deadlines first. In this work, when the AMA is available and $L \geq 1$ tasks are defined to be offloaded, we offload L tasks with the smallest deadlines. Whenever the local processing is available, we process the task with the smallest deadline. The proof for this optimality property is presented in Lemma 2.

Based on the strategy introduced above, given state \mathbf{s}_t and the offloading decision L_t , we define $\mathcal{C}(\mathbf{s}_t, L_t)$, the *expected instantaneous cost* incurred at time slot t by:

$$\mathcal{C}(\mathbf{s}_t, L_t) = p_u \mathcal{C}^A(\mathbf{s}_t, L_t) + (1 - p_u) \mathcal{C}^{\bar{A}}(\mathbf{s}_t) \quad (3)$$

in which

$$\mathcal{C}^A(\mathbf{s}_t, L_t) = C_o L_t + C_p \max(n_1^{(t)} - L_t, 0) \quad (4)$$

is the instantaneous cost when the AMA arrives and L_t tasks are offloaded. Similarly,

$$\mathcal{C}^{\bar{A}} = C_p n_1^{(t)} \quad (5)$$

is the cost when AMA is not available, thus, does not depend on the offloading decision L_t .

For a given time horizon T , an offloading *policy* π is a rule that determines the offloading parameter L_t for every state \mathbf{s}_t at t , i.e., $L_t = \pi(\mathbf{s}_t)$.

III. DYNAMIC PROGRAMMING FORMULATION

Consider the expected time-average cost given by

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\mathcal{C}(\mathbf{s}_t, L_t)]. \quad (6)$$

For the rest of this work, we will refer to the cost defined above as the *average cost*. We wish to determine the optimal offloading policy that minimizes the above average cost for a given time horizon.

Consider a state \mathbf{s} , in which the decision is to offload L tasks. Suppose a new task arrives with a deadline k . If local processing is available, the system will transit to a new state \mathbf{s}'_{Lk} ; if not, the new state will be \mathbf{s}''_{Lk} . These state definitions facilitate the analysis of the Dynamic Programming equation that will follow. They are formally defined in Eqs. (13) in this section. We need to introduce some additional, detailed notation to that effect.

Before doing this, we provide an example, let $\mathbf{s} = (0, 1, 2, 0, 1)$, the next event according to Fig. 2 is offloading, the AMA is available and $L = 2$ tasks with the most imminent deadlines are to be offloaded, therefore the intermediate state is $(0, 0, 1, 0, 1)$. The next event is deadline shifting to account for the new deadlines of the tasks in the next time slot, resulting in $(0, 1, 0, 1, 0)$. Following Fig. 2, we assume a new task with deadline 3 arrives, resulting in the intermediate state $(0, 1, 1, 1, 0)$. The next event is local processing, which processes the most imminent task, resulting in system state $\mathbf{s}'_{23} = (0, 0, 1, 1, 0)$.

Vector $\mathbf{o}(\mathbf{s}, L) = (o_1, \dots, o_N)$ represents the task offloading. Let d' be the deadline of the most imminent task after L tasks have been offloaded from \mathbf{s} . If $L < n_1$, $d' = 1$. Otherwise, d' satisfies the inequalities

$$L \leq \sum_{j=1}^{d'} n_j \text{ and } L > \sum_{j=1}^{d'-1} n_j. \quad (7)$$

Then, $\mathbf{o}(\mathbf{s}, L)$ is defined as follows.

- If $d' = 1$:

$$o_i = \begin{cases} L, & \text{if } i = 1, \\ 0, & \text{if } i \geq 2. \end{cases} \quad (8)$$

- If d' satisfies inequalities (7):

$$o_i = \begin{cases} n_i, & \text{if } i \leq d' - 1, \\ L - \sum_{j=1}^{d'-1} n_j, & \text{if } i = d', \\ 0, & \text{if } i \geq d' + 1. \end{cases} \quad (9)$$

Let $\mathbf{a}_k = (a_1, \dots, a_N)$ represent the task arrival vector defined as:

$$a_i = \begin{cases} 1 & , \text{ if } i = k, \\ 0 & , \text{ if } i \neq k. \end{cases} \quad (10)$$

Then, the intermediate state of the system after L most imminent tasks have been offloaded from \mathbf{s} , the deadline shifting has been performed, and the task arrival event has been realized, can be defined as follows:

$$\mathbf{s}^{inte} = f_{ds}(\mathbf{s} - \mathbf{o}(\mathbf{s}, L)) + \mathbf{a}_k \quad (11)$$

where the function $f_{ds}(\cdot)$ performs deadline shifting on the given vector. If $\mathbf{s} = (n_1, \dots, n_N)$, $f_{ds}(\mathbf{s})$ is an N -dimensional vector $(n_2, \dots, n_N, 0)$ in which the component n_1 is removed, representing task expiration.

Let $\mathbf{l} = (l_1, \dots, l_N)$ represent the local processing vector. If $\mathbf{s}^{inte} \neq (0, \dots, 0)$, we assume d is the deadline of the most imminent task in \mathbf{s}^{inte} , then elements of \mathbf{l} is given by:

$$l_i = \begin{cases} 1, & \text{ if } i = d, \\ 0, & \text{ if } i \neq d. \end{cases} \quad (12)$$

Otherwise, $\mathbf{l} = (0, \dots, 0)$.

Then the system state transition is defined as:

$$\mathbf{s}'_{Lk} = \mathbf{s}^{inte} - \mathbf{l}, \text{ and } \mathbf{s}''_{Lk} = \mathbf{s}^{inte}. \quad (13)$$

To this end, let $J_T(\mathbf{s})$ denotes the minimum average cost over T time slots for a given initial state \mathbf{s} . We have the following Dynamic Programming Equation.

$$J_T(\mathbf{s}) = \min_{L \in \mathbb{L}(\mathbf{s})} \{ \mathcal{C}(\mathbf{s}, L) + G_{T-1}(\mathbf{s}, L) \}, \quad (14)$$

with

$$G_T(\mathbf{s}, L) = \begin{cases} p_u G_T^A(\mathbf{s}, L) + (1 - p_u) G_T^{\bar{A}}(\mathbf{s}), & T \geq 1, \\ 0, & T = 0. \end{cases} \quad (15)$$

where

$$G_T^A(\mathbf{s}, L) = \mu \sum_{k=0}^N p_k J_T(\mathbf{s}'_{Lk}) + (1 - \mu) \sum_{k=0}^N p_k J_T(\mathbf{s}''_{Lk}) \quad (16)$$

is the average future cost given that the AMA arrives at the current time slot and L tasks are offloaded, and

$$G_T^{\bar{A}}(\mathbf{s}) = \mu \sum_{k=0}^N p_k J_T(\mathbf{s}'_{0k}) + (1 - \mu) \sum_{k=0}^N p_k J_T(\mathbf{s}'_{0k}) \quad (17)$$

is that without the AMA, and hence, does not depend on L .

Eq. (14) is equivalent to

$$J_{T+1}(\mathbf{s}) = p_u J_{T+1}^A(\mathbf{s}) + (1 - p_u) J_{T+1}^{\bar{A}}(\mathbf{s}) \quad (18)$$

in which

$$J_{T+1}^A(\mathbf{s}) = \min_{L \in \mathbb{L}(\mathbf{s})} \left\{ \mathcal{C}^A(\mathbf{s}, L) + G_T^A(\mathbf{s}, L) \right\} \quad (19)$$

and

$$J_{T+1}^{\bar{A}}(\mathbf{s}) = \mathcal{C}^{\bar{A}}(\mathbf{s}) + G_T^{\bar{A}}(\mathbf{s}). \quad (20)$$

To facilitate our analysis in the subsequent sections, we define the minimum average cost attained by offloading exactly L most imminent tasks from an initial state \mathbf{s} given the AMA's presence by

$$J_T^A(\mathbf{s}, L) = \mathcal{C}^A(\mathbf{s}, L) + G_T^A(\mathbf{s}, L). \quad (21)$$

We denote $\bar{\mathbf{s}}_{dL}$ the state obtained by offloading, from \mathbf{s} , L most imminent tasks having deadline greater than or equal to d . The following example is to provide more intuition about this notation.

Example 1: Given state $\mathbf{s} = (0, 5, 6, 7, 8)$, the state $\bar{\mathbf{s}}_{37}$ is obtained by offloading 7 most imminent tasks starting from deadline 3, thus, $\bar{\mathbf{s}}_{37} = (0, 5, 0, 6, 8)$. Now, if we offload 7 most imminent tasks starting from deadline 5, the resulted state would be $\bar{\mathbf{s}}_{57} = (0, 5, 6, 7, 1)$.

We have mentioned in the previous section that whenever offloading is possible, and L tasks need to be offloaded from a given state \mathbf{s} , it is optimal to offload L most imminent tasks. This offloading behaviour results in state $\bar{\mathbf{s}}_{1L}$ which is always associated with $d = 1$. However, the properties we studied in Subsection V-B are associated with a general context where tasks can be offloaded starting from an arbitrary deadline d . Therefore, for consistency, we introduce a general notation $\bar{\mathbf{s}}_{dL}$ where d is used in the subscript instead of 1.

For the rest of this article, the terms *offloading states* and *non-offloading states* will be used. Therefore, they are specified in the following definition.

Definition 1: For a given state \mathbf{s} and a time horizon T , \mathbf{s} is called an offloading state if the associated optimal offloading decision is a positive integer. State \mathbf{s} is called a non-offloading state if the associated optimal offloading decision is 0.

IV. COMPUTATIONAL LOAD REDUCTION

A. Reduced State Space

For a given N -dimensional state vector \mathbf{s} , since there are at most $N - 1$ tasks that can be processed in N time slots. This is because the deadline shifting happens at the beginning of every time slot as presented in Fig 2. Therefore, tasks having deadline 1 cannot be processed. This means that there might be certain tasks that are *guaranteed to expire* if not offloaded within the next N time slots; we will call such tasks *excessive tasks*. We define the *reduced states* as the ones having no excessive tasks.

In the remainder of this sub-section we will provide a characterization and properties of reduced states. For such a state $\mathbf{s} = (n_1, \dots, n_N)$ and from Fig. 2, all tasks having deadline 1 will expire if not offloaded, hence, we must have $n_1 = 0$. Next, at most one task can be processed in the next slot, therefore, we must have $n_1 + n_2 \leq 1$. Subsequently, at most two tasks can be processed in the next two time slots, leading to $n_1 + n_2 + n_3 \leq 2$, otherwise at least one task will expire after two slots, and so on. Finally, at most $N - 1$ tasks can be processed in N time slots including the current one, so we must have $\sum_{i=1}^N n_i \leq N - 1$. The definition for a reduced states follows.

Definition 2: A state $\mathbf{s}_r = (n_1^r, \dots, n_N^r)$ is a reduced state if and only if the following inequalities hold:

$$\sum_{i=1}^m n_i^r \leq m - 1, m = 1, 2, \dots, N. \quad (22)$$

As elements of a reduced state vector is bounded, the number of reduced vectors is finite. Moreover, the next lemma states that the number of reduced state vectors is equal to the Catalan number [11].

Lemma 1: The number of reduced states having dimension N is finite and equals to the Catalan number $C_N = \binom{2N}{N} / (N + 1)$.

Proof: Please see Appendix X-A.

Abusing the notation slightly, for the sake of simplicity, we can associate a corresponding reduced state $\mathbf{s}_r = (n_1^r, \dots, n_N^r)$ for any given state $\mathbf{s} = (n_1, \dots, n_N)$, in the infinite state

space, using Algorithm IV.1. For $\mathbf{s} = (n_1, \dots, n_N)$, it can be seen from Eq. (14) and Eqs. (3)-(5) that $n_i, i \geq 2$ do not contribute to the cost $J_1(\mathbf{s})$. Similarly, $n_i, i \geq 3$ do not contribute to the cost $J_2(\mathbf{s})$. Observe, therefore that in general, tasks having deadline greater than the considered time horizon will not contribute to the cost in Eq. (14). Therefore, these tasks will not be considered, which is reflected by line 4 of this algorithm.

Algorithm IV.1 Derivation of Reduced States

```

1: Input:  $\mathbf{s} = (n_1, n_2, \dots, n_N)$ ,  $N$ ,  $T$ .
2: Output:  $\mathbf{s}_r$ ,  $L_g$ .
3: Initialize:  $t_{\text{idle}} \leftarrow 0$ ,  $L_g \leftarrow 0$ ,  $\tilde{\mathbf{s}} \leftarrow (\tilde{n}_1, \dots, \tilde{n}_N)$  where  $\tilde{n}_i \leftarrow n_i, i \leq T$ , and  $\tilde{n}_i \leftarrow 0, i > T$ .
4:  $n_i \leftarrow 0$ , for  $i > \min(N, T)$ .
5: for  $i = 1, 2, \dots, N$  do
6:   for  $j = 1, 2, \dots, \min(N, T)$  do
7:      $a \leftarrow \sum_{d=1}^j \tilde{n}_d - \max(n_i - i + t_{\text{idle}} + 1, 0)$ .
8:     if  $a \geq 0$  then
9:       if  $j \geq 2$  then
10:         $\tilde{n}_d \leftarrow 0, d = 1, \dots, j - 1$ .
11:         $\tilde{n}_j \leftarrow \tilde{n}_j - a$ .
12:    $L_g \leftarrow L_g + \max(n_i - i + t_{\text{idle}} + 1, 0)$ .
13:    $t_{\text{idle}} \leftarrow \sum_{d=1}^i \tilde{n}_d$ .
14:  $\mathbf{s}_r \leftarrow$  Offloading  $L_g$  most imminent tasks from  $\mathbf{s}$ .
15: return  $\mathbf{s}_r$ ,  $L_g$ .
```

Determining the reduced states is the first step in reducing the computational burden of the DP equation (14). The next step is the determination of a new concept know as the lean states, which also constitute a finite set and are directly used to compute the optimal cost $J_T(\mathbf{s})$ according to the forthcoming Proposition 1.

B. Lean State Space

The definition of lean states is given as below:

Definition 3: For a given state $\mathbf{s} = (n_1, \dots, n_N)$, we call $\mathbf{s}_r = (n_1^r, \dots, n_N^r)$ a reduced state obtained from \mathbf{s} following Algorithm IV.1. We define the parameters $\gamma_j, j = 1, \dots, N$ as follows:

$$\gamma_j = \begin{cases} 0 & , \text{ if } j = 1 \text{ or } j > \min(N, T), \\ \min\{n_j, j - 1 - \sum_{i=1}^{j-1} \gamma_i\} & , \text{ otherwise.} \end{cases} \quad (23)$$

Then, the lean state $\mathbf{s}_m = (n_1^m, \dots, n_N^m)$ corresponding to \mathbf{s} is given by

$$n_i^m = \max\{\gamma_i, n_i^r\}, \quad i = 1, 2, \dots, N. \quad (24)$$

The relation in the minimum average cost of a given state \mathbf{s} and that of its corresponding lean state is presented in Proposition 1.

Proposition 1: Given state $\mathbf{s} = (n_1, \dots, n_N)$ and its lean state $\mathbf{s}_m = (n_1^m, \dots, n_N^m)$. Let P_t be the probability that the AMA is available for the first time at time slot t . Then, the following equality holds.

$$J_T(\mathbf{s}) = J_T(\mathbf{s}_m) + C_{g2m} \quad (25)$$

where

$$C_{g2m} = C_o \sum_{i=1}^N (n_i - n_i^m) \sum_{j=0}^{i-1} P_{t=j} + C_p \sum_{i=1}^{N-1} P_{t=i} \sum_{j=1}^i (n_j - n_j^m) + C_p P_{t \geq N} \sum_{j=1}^N (n_j - n_j^m)$$

with the probabilities $P_{t=j}$ and $P_{t \geq N}$ are computed by

$$P_{t=j} = p_u (1 - p_u)^j,$$

$$P_{t \geq N} = (1 - p_u)^N.$$

Proof: Please see Appendix XI-A.

As the lean state space is finite, if $J_T(\mathbf{s}_m)$ is computed for all lean states via the DP equation (14), $J_T(\mathbf{s})$ can be computed for every \mathbf{s} via the simple algebraic manipulation of equation (25).

V. PROPERTIES OF OPTIMAL POLICY AND COST FUNCTION

A. Optimality of the Most-Imminent Offloading Method

The first property of an optimal offloading policy is offloading the most imminent tasks first. This property will be proved based on another property presented in the next proposition.

Proposition 2: Given state $\mathbf{s} = (n_1, n_2, \dots, n_N) \neq (0, \dots, 0)$. Assuming that d is the deadline of the most imminent task of \mathbf{s} , we define the set $\mathbb{S}_{pp}(\mathbf{s})$ as follows. If $\tilde{\mathbf{s}} = (\tilde{n}_1, \dots, \tilde{n}_N) \in \mathbb{S}_{pp}(\mathbf{s})$, state $\tilde{\mathbf{s}}$ satisfies the following conditions:

- *If $N = 1$: $\tilde{\mathbf{s}} \equiv \mathbf{s}$.*
- *If $N \geq 2$: either $\tilde{\mathbf{s}} \equiv \mathbf{s}$, or $\tilde{\mathbf{s}}$ is defined by*

$$\tilde{n}_i = \begin{cases} n_d - 1, & \text{if } i = d, \\ n_j + 1, & \text{if } i = \tilde{d} \text{ for a deadline } \tilde{d} \in \{d + 1, \dots, N\}, \\ n_i, & \text{otherwise.} \end{cases} \quad (26)$$

Then, the following inequality holds:

$$J_T(\mathbf{s}) \geq J_T(\tilde{\mathbf{s}}), \text{ for all } T, \mathbf{s}, \tilde{\mathbf{s}} \in \mathbb{S}_{pp}(\mathbf{s}). \quad (27)$$

Proof: Please see Appendix XI-B.

As a result of Proposition 2, the first property of the optimal policy can be proved, and formally presented in Lemma 2.

Lemma 2: When processing is possible, it is optimal to process the most imminent task. When it is optimal to offload L tasks, the L most imminent tasks should be offloaded.

Proof: Please see Appendix X-B.

B. Convexity of the Minimum average cost with respect to the Offloading Decision

For a given time horizon T , and an initial state $\mathbf{s} = (n_1, \dots, n_N)$, we define the following function

$$F^{\bar{A}}(T, \mathbf{s}, d, L) = J_T^{\bar{A}}(\bar{\mathbf{s}}_{dL}) + LC_o, \quad (28)$$

in which we recall that $\bar{\mathbf{s}}_{dL}$ is the state obtained by offloading, from \mathbf{s} , L most imminent tasks having deadline greater than or equal to d . Function $F^{\bar{A}}(T, \mathbf{s}, d, L)$ is characterized by parameters T, \mathbf{s} , and d . This function takes L as variable. We define the domains of $F^{\bar{A}}(T, \mathbf{s}, d, L)$ as follows.

$$\mathbb{L}_1(\mathbf{s}) = \left\{ n_1, \dots, \sum_{i=1}^N n_i \right\}, \quad (29)$$

$$\mathbb{L}_d(\mathbf{s}) = \left\{ 0, 1, \dots, \sum_{i=d}^N n_i \right\}, \text{ for } d = 2, \dots, N. \quad (30)$$

$\mathbb{L}_d(\mathbf{s})$ can be interpreted as the set of valid offloading decisions associated with state \mathbf{s} and deadline d . In the definition of $\mathbb{L}_1(\mathbf{s})$, because all tasks having deadline 1 are excessive tasks and must be offloaded, the optimal offloading decision must be conveyed in the set $\{n_1, \dots, \sum_{i=1}^N n_i\}$. Hence, $L < n_1$ can be removed from the domains of $F^{\bar{A}}(T, \mathbf{s}, d, L)$ when $d = 1$.

Example 2: As an example, let us consider a 3-dimensional state vector $\mathbf{s} = (2, 3, 4)$. In this example, we have

$$\mathbb{L}_1(\mathbf{s}) = \{2, 3, \dots, 9\}, \quad (31)$$

$$\mathbb{L}_2(\mathbf{s}) = \{0, 1, \dots, 7\}, \quad (32)$$

$$\mathbb{L}_3(\mathbf{s}) = \{0, 1, \dots, 4\}. \quad (33)$$

We have the following convexity property stated in the next lemma.

Lemma 3: For every given time horizon T , an initial state \mathbf{s} , and a deadline d , the function $F^{\bar{A}}(T, \mathbf{s}, d, L)$ as defined in Eq. (28) is a discrete convex function with respect to $L \in \mathbb{L}_d(\mathbf{s})$ where $\mathbb{L}_d(\mathbf{s})$ is defined in Eqs. (29) and (30).

Proof: Please see Appendix X-C.

The next lemma states the relation between function $F^{\bar{A}}(T, \mathbf{s}, d, L)$ and the optimal offloading decision associated with state \mathbf{s} and a time horizon T .

Lemma 4: Assuming that the function $F^{\bar{A}}(T, \mathbf{s}, d, L)$ attains its minimum at L^* for $d = 1$, then, L^* is the optimal offloading decision of \mathbf{s} for a time horizon T .

Proof: Please see Appendix X-D.

Other important properties of the optimal offloading decisions will be presented in the next subsection based on the concept of *adjacent states*.

C. Adjacent States

The goal of this subsection is introducing the concept of adjacency among states, and related properties. These properties facilitate the design of the optimal policy presented later on. The definition of adjacent states is given below.

Definition 4: Consider a state $\mathbf{s} = (n_1, \dots, n_N) \neq (0, \dots, 0)$, with d as the smallest deadline such that $n_d > 0$. Then, state $\mathbf{s}_a = (n_1^a, \dots, n_N^a)$ is an adjacent state to \mathbf{s} if there exists a

deadline $j \in \{1, \dots, d\}$ such that:

$$n_i^a = \begin{cases} n_i + 1, & \text{if } i = j, \\ n_i, & \text{if } i \neq j. \end{cases} \quad (34)$$

If a state \mathbf{s}_a has only one task with arbitrary deadline, it is adjacent to state $(0, \dots, 0)$.

Let us consider the following two examples.

Example 3: In the first example, we assume that a state $\mathbf{s} = (0, 0, 1, 4, 4)$ is given in which the deadline of the most imminent task in \mathbf{s} is 3. Therefore, an adjacent state \mathbf{s}_a of \mathbf{s} can be obtained by adding a task with deadline less than or equal to 3, e.g., $\mathbf{s}_a = (0, 1, 1, 4, 4)$.

In the second example, assuming that $\mathbf{s}_a = (0, 2, 1, 3, 3)$. A state \mathbf{s} for which \mathbf{s}_a is adjacent, can be obtained by offloading the most imminent task in \mathbf{s}_a , i.e., $\mathbf{s} = (0, 1, 1, 3, 3)$.

We denote by $\mathbb{S}_{adj}(\mathbf{s})$ the set of all adjacent states of \mathbf{s} . For a given time horizon, the optimal offloading decision of \mathbf{s} can be inferred from that of $\mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$ and vice versa, as described in Theorem 1.

Theorem 1: Given two states $\mathbf{s}, \mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$, and a time horizon T . We call L^* and L_a^* the optimal offloading decision of \mathbf{s} and \mathbf{s}_a , respectively, for the time horizon T . We have the following relations:

- 1) If $L_a^* \geq 1$, $L^* = L_a^* - 1$.
- 2) If $L_a^* = 0$, $L^* = 0$.
- 3) If $L^* \geq 1$, $L_a^* = L^* + 1$.

Proof: Please see Appendix IX-A.

D. Offloading and Non-Offloading Conditions

We have defined the notion of offloading and non-offloading states in Definition 1. In this subsection, we mathematically identify the conditions for a state to be an offloading and non-offloading state for a given time horizon. This is formally stated in Proposition 3.

Proposition 3: Assuming that two states $\mathbf{s}, \mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$, and a time horizon T is given. \mathbf{s}_a is a non-offloading state if and only if the following inequality holds

$$J_T(\mathbf{s}_a) - J_T(\mathbf{s}) < C_o. \quad (35)$$

Otherwise, \mathbf{s}_a is an offloading state.

Proof: Please see Appendix XI-C.

Assume we are given a state \mathbf{s} and a time horizon T with L^* as the optimal offloading decision. We recall that $\bar{\mathbf{s}}_{1L}$ is the state that resulted by removing the first L most imminent tasks from \mathbf{s} . Then, the next property of the optimal offloading policy is stated as follows.

Theorem 2: The optimal offloading decision is the smallest offloading decision L such that $\bar{\mathbf{s}}_L$ is a non-offloading state.

Proof: Please see Appendix IX-B.

VI. OPTIMAL OFFLOADING POLICY

A. Optimal Policy Description

Given an initial state \mathbf{s} and a time horizon T , whenever local processing is available to process a task, the most imminent task will be processed. When a AMA is present, the optimal policy consists of two steps.

Step 1. Tasks are offloaded from \mathbf{s} following Algorithm IV.1 to reach a reduced state \mathbf{s}_r . Moreover the value of L_g , the number of offloaded tasks is determined.

Step 2. In this step, the DP equation is solved recursively and

$$L_r = \arg \min_{L \in \mathbb{L}(\mathbf{s}_r)} \{ \mathcal{C}(\mathbf{s}_r, L) + G_{T-1}(\mathbf{s}_r, L) \}$$

tasks are offloaded from \mathbf{s}_r . Note that as described in equation (25), the recursion involves evaluation of only a finite number states in each one of the terms $J_{T-1}(\cdot)$, $J_{T-2}(\cdot)$ etc. for any state \mathbf{s} . The optimal number of tasks that are offloaded from \mathbf{s} is $L^* = L_g + L_r$.

The computational load required to solve the recursive DP equation (14) can be further reduced by the following strategy. Every time the optimal decision L_r associated with a reduced state \mathbf{s}_r is computed for a given T , the triplet (\mathbf{s}_r, L^*, T) is saved. This can be done for all the reduced states, as the number of reduced states is finite. Then, for every \mathbf{s}_r obtained from Step 1 for a given T , the corresponding L^* can be retrieved instantly.

By exploiting the properties presented in Theorem 1, the computational burden can be further reduced as follows. Let us consider a sequence of adjacent states: $\mathbf{s}_1, \dots, \mathbf{s}_i, \dots$ in which $\mathbf{s}_{i+1} \in \mathbb{S}_{adj}(\mathbf{s}_i)$. Assume the optimal decision of state $\mathbf{s}_i, i \geq 1$ is known, and denoted by L_i^* . From Theorem 1, the optimal decisions L_{i-u}^* of states $\mathbf{s}_{i-u}, u = 1, \dots, i-1$, can be inferred as follows

$$L_{i-u}^* = \max(L_i^* - u, 0), u = 1, \dots, i-1. \quad (36)$$

T	p_u	C_p	C_o	μ	p_0	N	$p_i \forall i: 1 \leq i \leq N$
1,000	0.7	3	1	0.7	0.5	3	1/6

TABLE I: System Parameters for Section VII-A

In the case when $L_i^* \geq 1$ for state \mathbf{s}_i , the optimal decision for states $\mathbf{s}_{i+v}, v = 1, 2, \dots$ are computed by

$$L_{i+v}^* = L_i^* + v, v = 1, 2, \dots \quad (37)$$

In general, the optimal offloading decisions of all the states \mathbf{s}_{i+v} and \mathbf{s}_{i-u} mentioned above can be obtained without relying on the DP equation (14).

In the next section, we provide numerical results to verify the results presented in this paper thus far.

VII. NUMERICAL RESULTS

In this section, we present numerical examples that help visualize some of the properties and equations as well as show the memory savings of the numerical computations of the DP equation (14) with the aid of Eq. (25) described in Sec. IV-B. Some of the examples utilize different sets of parameters to illustrate the performance of the system under different parameter configurations.

A. Optimal Offloading Decision Visualization

In this example, we illustrate the idea presented in Theorem 2 visually for a system with the dimension of state vector to be $N = 3$, i.e. $\mathbf{s} = (n_1, n_2, n_3)$. We represent states as coordinates in a 3-D state space. Then, for $n_3 = 0, 1, 2, 3$, we consider the 2-D slices of this space and depict them as Figs. 3a-3c correspondingly. We use the system parameters described in Table I. The cases when $n_3 \geq 3$ would result in similar figures as Fig. 3c except that the optimal offloading decision for each offloading state would increase by $n_3 - 2$. In these figures, red dots represent offloading states, and black dots represent non-offloading states. From states with component $n_2 \geq 1$ in Fig. 3a, such as $(0, 2, 0)$, $(1, 2, 0)$, $(1, 1, 0)$, $(2, 1, 0)$, etc., we can reach the non-offloading state $(0, 1, 0)$ with a smaller number of offloaded tasks than state $(0, 0, 0)$. A similar argument applies for states with component $n_2 = 0$, like $(1, 0, 0)$, $(2, 0, 0)$, etc., whose “nearest” non-offloading state is $(0, 0, 0)$.

p_u	C_p	C_o	μ	p_0	$p_i \forall i: 1 \leq i \leq N$
0.5	3	1	0.5	0.5	0.1

TABLE II: System Parameters for Fig. 4a

p_u	C_p	C_o	μ	p_0	$p_i \forall i: 1 \leq i \leq N$
0.4	4	1	0.3	0.3	0.14

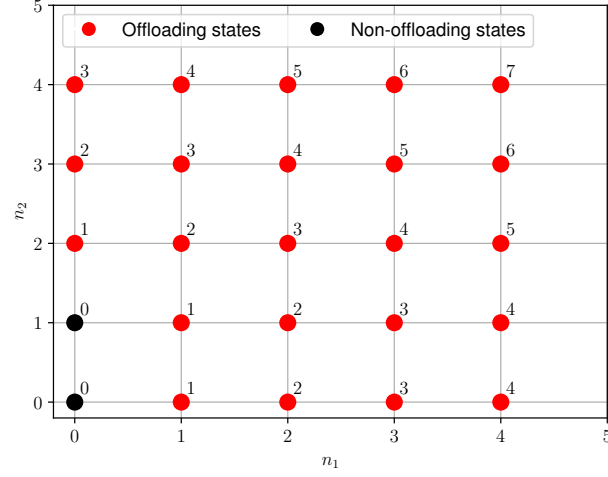
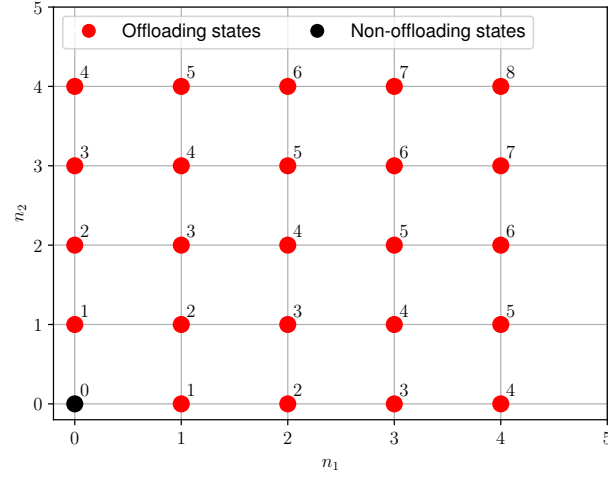
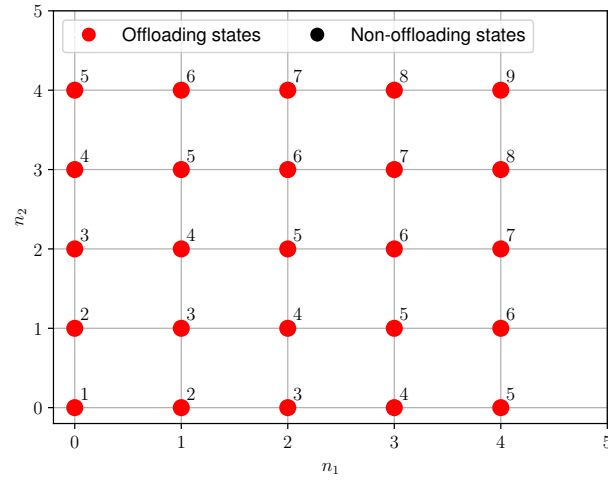
TABLE III: System Parameters for Fig. 4b

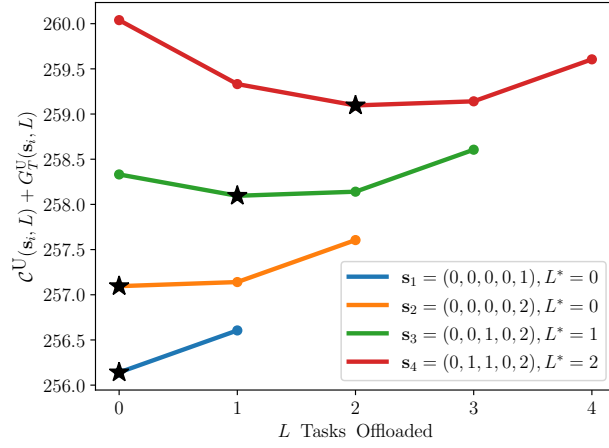
In Fig. 3b, only the state $(0, 0, 1)$ is non-offloading. The optimal offloading decisions of all the offloading states shown are the smallest number of most imminent tasks to be offloaded to reach state $(0, 0, 1)$. Fig. 3c does not have any non-offloading state. For example, the optimal offloading decision for the state $(0, 0, 3)$ is 1 to reach the non-offloading state $(0, 0, 2)$ which is shown in Fig. 3b.

B. Optimal Offloading Decisions for Adjacent States

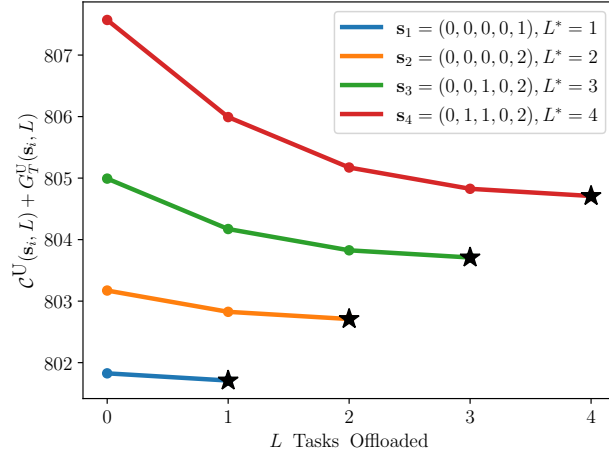
In Fig. 4a-4c we visualize the results of equations (36), (37) and Theorem 1. The figures show the optimal offloading decisions marked with a star symbol in adjacent states. For these examples, all figures have a state size $N = 5$ over a time horizon $T = 1,000$. Fig. 4a, 4b and 4c use the parameters listed in Tables II, III and IV, respectively. On the vertical axis, we graph the minimum cost $\mathcal{C}^A(\mathbf{s}_i, L) + G_T^A(\mathbf{s}_i, L)$ attained by offloading L most imminent tasks from state \mathbf{s}_i given that the AMA is available. This is introduced in Eq. (21).

In these three figures, for $i = 1, 2, 3, 4$, we consider the states $\mathbf{s}_1 = (0, 0, 0, 0, 1)$, $\mathbf{s}_2 = (0, 0, 0, 0, 2)$, $\mathbf{s}_3 = (0, 0, 1, 0, 2)$, and $\mathbf{s}_4 = (0, 1, 1, 0, 2)$. These states are chosen such that \mathbf{s}_i is adjacent to \mathbf{s}_{i+1} , $i = 1, 2, 3$. The presented results indicate that the optimal offloading decision of a state differs from that of its adjacent state by 1, or both are capped at 0, as Eqs. (36)-(37) and Theorem 1 suggest. For example, in Fig. 4a, the optimal decision of \mathbf{s}_4 is 2, and that of \mathbf{s}_3 is 1, hence, the difference is 1. The same observation applies for the pair \mathbf{s}_3 and \mathbf{s}_2 . The optimal decision of \mathbf{s}_2 is 0, therefore, that of \mathbf{s}_1 is also 0. Figs 4b and 4c present the same properties.

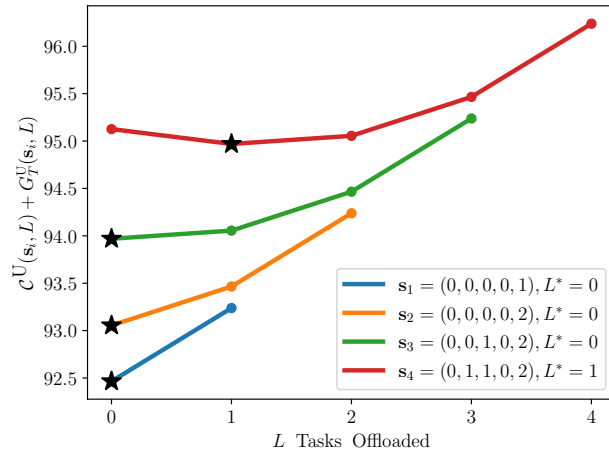
(a) $n_3 = 0$ (b) $n_3 = 1$ (c) $n_3 = 2$ Fig. 3: Offloading states for varying n_3 values



(a) Adjacent states using parameters in Table II



(b) Adjacent states using parameters in Table III



(c) Adjacent states using parameters in Table IV

Fig. 4: Visual interpretation of Eqs. (36)-(37) and Theorem 1.

p_u	C_p	C_o	μ	p_0	$p_i \forall i: 1 \leq i \leq N$
0.7	2	1	0.6	0.6	0.08

TABLE IV: System Parameters for Fig. 4c

C. Memory Savings Using Equation (25)

In order to numerically compute the DP equation (14), we store the computed value of $J_T(\mathbf{s})$ in memory for a given arbitrary state \mathbf{s} and time horizon T . By using Eq. (25), the size of memory required is reduced. This is because saving $J_T(\cdot)$ values is only required for lean states \mathbf{s}_m where the number of lean states is smaller than that of “generic” states. Note that these savings are achieved at the expense of calculating the term C_{g2m} in Eq. (25).

In Fig. 5a, 5b, and Fig. 5c, we show the resulting difference in the number of $J_T(\cdot)$ values saved for 2 cases when $N = 5$, $N = 4$ and $N = 3$, respectively. Note that different parameters other than the system parameters N and T will not affect the memory savings. In the first case, Eq. (14) is used, while in the second case, Eq. (14) is used with the aid of Eq. (25). The line in blue represents the values saved using only Eq. (14), while the line in red utilizes both Eqs. (14) and (25).

VIII. CONCLUSION

In this work, we studied a mobile edge computing system with dynamic user demand. In the context of an optimal stochastic control framework for serving user tasks, we considered the following features: tasks with firm deadlines, their random offloading to a remote server (AMA) or their processing by a local server (BS) with intermittent service. We considered an expected time-average cost over a finite time horizon and formulated a Dynamic Programming problem towards the minimization of this cost. In order to tackle the “Curse of Dimensionality”, we studied important characteristics of the optimal policy and reduced the computational load for its calculation. In particular, we proved that the DP equation can be evaluated for every given state (in the infinite state space of our model) by considering a specific finite space called lean state space. Further reduction in the computational load was achieved by using the concept of “adjacent states”. This allowed us to evaluate the optimal cost for all such states from knowledge of the cost in only one state. Finally, based on these properties, we described an optimal task offloading policy.

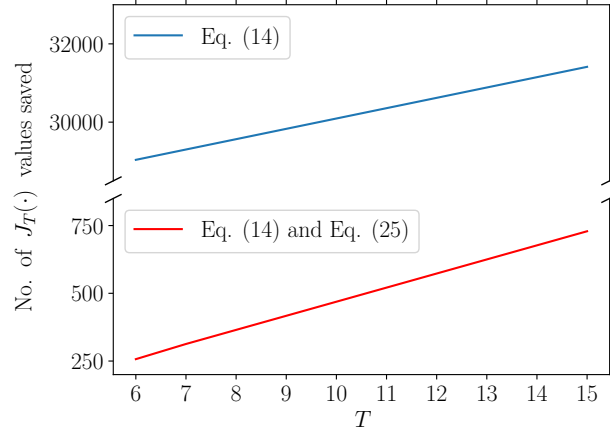
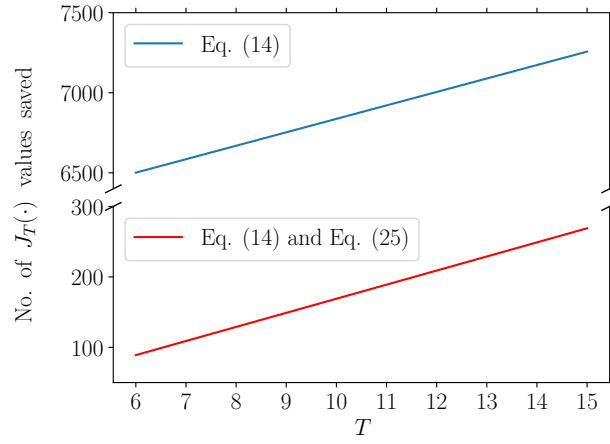
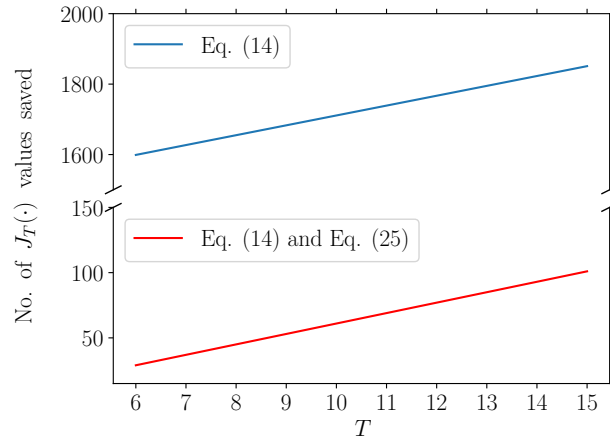
(a) Memory savings with $N = 5$ (b) Memory Saving with $N = 4$ (c) Memory Saving with $N = 3$

Fig. 5: Memory Savings using Eq. (25)

IX. APPENDIX A: PROOFS OF THEOREMS

For a given initial state \mathbf{s} , and a time horizon T , we recall that $\bar{\mathbf{s}}_{1L}$ is the state obtained by offloading L most imminent tasks from state \mathbf{s} . Combining this definition with that of $J_T^A(\mathbf{s}, L)$ in Eq. (21), we have the following relation

$$J_T^A(\mathbf{s}, L) = J_T^A(\bar{\mathbf{s}}_{1L}, 0) + LC_o, \text{ for } L \in \mathbb{L}(\mathbf{s}) \quad (38)$$

which can be explained as follows. We note that by offloading L most imminent tasks from \mathbf{s} , we pay a cost LC_o , and arrive at state $\bar{\mathbf{s}}_{1L}$. Therefore, if we wish to describe the offloading of L tasks on the left hand side of Eq. (38), this is equivalent to removing the L most imminent tasks from state \mathbf{s} to reach state $\bar{\mathbf{s}}_{1L}$, and offloading 0 task from $\bar{\mathbf{s}}_{1L}$. Finally, we further add the offloading cost LC_o to the overall cost. The right hand side of Eq. (38) describes this idea.

Since the minimum average cost when offloading no task is the same regardless of the presence of the AMA's availability, i.e.,

$$J_T^A(\bar{\mathbf{s}}_{1L}, 0) = J_T^{\bar{A}}(\bar{\mathbf{s}}_{1L}), \text{ for } L \in \mathbb{L}(\mathbf{s}), \quad (39)$$

we have

$$J_T^A(\mathbf{s}, L) = J_T^{\bar{A}}(\bar{\mathbf{s}}_{1L}) + LC_o, \text{ for } L \in \mathbb{L}(\mathbf{s}). \quad (40)$$

Furthermore, considering two offloading decisions L_1 and L_2 where $L_1 + L_2, L_1, L_2 \in \mathbb{L}(\mathbf{s})$, we note that offloading $L_1 + L_2$ tasks from the original state \mathbf{s} would result in the same state that is obtained by offloading L_2 tasks from $\bar{\mathbf{s}}_{1L_1}$. An example for this point is as follows:

Example 4: Assume $\mathbf{s} = (0, 1, 2, 2, 3)$, $L_1 = 2, L_2 = 3$, then, the resulting states by offloading $L_1 + L_2 = 5$ and $L_1 = 2$ tasks from \mathbf{s} , respectively, are

$$\bar{\mathbf{s}}_{1(L_1+L_2)} = (0, 0, 0, 0, 3), \quad (41)$$

$$\bar{\mathbf{s}}_{1L_1} = (0, 0, 1, 2, 3). \quad (42)$$

Now, offloading $L_2 = 3$ tasks from $\bar{\mathbf{s}}_{1L_1}$ would give us the same state as in Eq. (41).

From Eq. (40), we have

$$J_T^A(\mathbf{s}, L_1 + L_2) = J_T^{\bar{A}}(\bar{\mathbf{s}}_{1(L_1+L_2)}) + (L_1 + L_2) C_o, \quad (43)$$

$$J_T^A(\bar{\mathbf{s}}_{1L_1}, L_2) = J_T^{\bar{A}}(\bar{\mathbf{s}}_{1(L_1+L_2)}) + L_2 C_o. \quad (44)$$

The above two expressions result in the following one:

$$J_T^A(\mathbf{s}, L_1 + L_2) = J_T^A(\bar{\mathbf{s}}_{1L_1}, L_2) + L_1 C_o. \quad (45)$$

A. Proof of Theorem 1

From Lemma 3, we have that function $F^{\bar{A}}(T, \mathbf{s}, d, L)$ is discrete convex with respect to L for every given T , d , and $\mathbf{s} = (n_1, \dots, n_N)$. Let us consider the case when $d = 1$, and we call $L^* \in \mathbb{L}_1(\mathbf{s})$ the value at which $F^{\bar{A}}(T, \mathbf{s}, 1, L)$ attains its minimum. From Lemma 4, L^* is also the optimal offloading decision for \mathbf{s} . Hence,

$$J_T^A(\mathbf{s}, L) \geq J_T^A(\mathbf{s}, L^*), \text{ for all } L \in \mathbb{L}_1(\mathbf{s}), \quad (46)$$

Let us consider the following cases:

- If $L^* \geq 1$ which means that $\mathbf{s} \neq (0, \dots, 0)$, the set of inequalities (46) can be re-written as

$$J_T^A(\mathbf{s}, 1 + L - 1) \geq J_T^A(\mathbf{s}, 1 + L^* - 1), \text{ for all } L \in \mathbb{L}_1(\mathbf{s}) \setminus \{0\}. \quad (47)$$

Applying Eq. (45) with $L_1 = 1$, $L_2 = L - 1$ to the left-hand side of Ineqs. (47), and with $L_2 = L^* - 1$ to the right-hand side of Ineqs. (47), we have

$$J_T^A(\bar{\mathbf{s}}_{11}, L - 1) \geq J_T^A(\bar{\mathbf{s}}_{11}, L^* - 1), \text{ for all } L, L^* \in \mathbb{L}_1(\mathbf{s}) \setminus \{0\}, \quad (48)$$

in which we recall that $\bar{\mathbf{s}}_{11}$ is obtained by offloading the most imminent task from \mathbf{s} , thus, $\mathbf{s} \in \mathbb{S}_{adj}(\bar{\mathbf{s}}_{11})$.

Since $L, L^* \in \mathbb{L}_1(\mathbf{s}) \setminus \{0\}$, we have $L - 1, L^* - 1 \in \mathbb{L}_1(\bar{\mathbf{s}}_{11})$ ¹. Therefore, Ineqs. (48) suggests that $L^* - 1$ is the optimal offloading decision for $\bar{\mathbf{s}}_{11}$. This proves the first point of Theorem 1.

- If $L^* = 0$, i.e., \mathbf{s} is a non-offloading state. Then, if \mathbf{s} has only one task, we have $\bar{\mathbf{s}}_{11} \equiv (0, \dots, 0)$. Therefore, $\bar{\mathbf{s}}_{11}$ is a non-offloading state trivially.

If \mathbf{s} has at least 2 tasks. From the convexity of the function $F^{\bar{A}}(T, \mathbf{s}, d, L) = J_T^{\bar{A}}(\bar{\mathbf{s}}_{dL}) + LC_o$, and the condition that $L^* = 0$, we have

$$\begin{aligned} J_T^{\bar{A}}(\mathbf{s}) &\leq J_T^{\bar{A}}(\bar{\mathbf{s}}_{11}) + C_o, \text{ for all } L \in \mathbb{L}_1(\mathbf{s}) \setminus \{0, 1\}, \\ J_T^{\bar{A}}(\bar{\mathbf{s}}_{11}) + C_o &\leq J_T^{\bar{A}}(\bar{\mathbf{s}}_{1L}) + LC_o, \text{ for all } L \in \mathbb{L}_1(\mathbf{s}) \setminus \{0, 1\}. \end{aligned} \quad (49)$$

¹Note: The set $\mathbb{L}_1(\mathbf{s})$ is the same as $\mathbb{L}_1(\bar{\mathbf{s}}_{11})$ except that $\mathbb{L}_1(\mathbf{s})$ contains an element which is the total number of tasks in \mathbf{s} , while $\mathbb{L}_1(\bar{\mathbf{s}}_{11})$ does not contain that element.

where the two inequalities in Ineq. (49) are from the optimality of L^* , and the convexity of cost functions proven in Subsec. X-C, respectively.

Applying Eq. (39) to the second inequality of Ineqs. (49), we have

$$J_T^A(\bar{\mathbf{s}}_{11}, 0) + C_o \leq J_T^A(\bar{\mathbf{s}}_{1L}, 0) + LC_o, \text{ for all } L \in \mathbb{L}_1(\mathbf{s}) \setminus \{0, 1\}. \quad (50)$$

Using Eq. (45) with $L_1 = 1$ and $L_2 = L - 1$ gives us

$$J_T^A(\mathbf{s}, L) = J_T^A(\bar{\mathbf{s}}_{11}, L - 1) + C_o. \quad (51)$$

Also, using Eq. (45) with $L_1 = L$ and $L_2 = 0$ yields

$$J_T^A(\mathbf{s}, L) = J_T^A(\bar{\mathbf{s}}_{1L}, 0) + LC_o. \quad (52)$$

From Eqs. (51) and (52), we have

$$J_T^A(\bar{\mathbf{s}}_{11}, L - 1) + C_o = J_T^A(\bar{\mathbf{s}}_{1L}, 0) + LC_o. \quad (53)$$

Combining Eq. (53) with Ineqs. (50) gives the following set of inequalities:

$$J_T^A(\bar{\mathbf{s}}_{11}, 0) \leq J_T^A(\bar{\mathbf{s}}_{11}, L - 1), \text{ for all } L \in \mathbb{L}_1(\mathbf{s}) \setminus \{0, 1\}. \quad (54)$$

By replacing $L - 1$ with \tilde{L} in the above inequalities, we have

$$J_T^A(\bar{\mathbf{s}}_{11}, 0) \leq J_T^A(\bar{\mathbf{s}}_{11}, \tilde{L}), \text{ for all } \tilde{L} \in \mathbb{L}_1(\bar{\mathbf{s}}_{11}) \setminus \{0\}, \quad (55)$$

indicating that $\bar{\mathbf{s}}_{11}$ is a non-offloading state. This proves the second point of Theorem 1.

Finally, from the first two points of Theorem 1, we can conclude that, for given time horizon T , if $L^* \geq 1$ is the optimal offloading decision for state \mathbf{s} , $L_a^* = L^* + 1$ is optimal for every state $\mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$. This is because if $L_a^* \neq L^* + 1$ and $L_a^* \geq 1$, from the first point of Theorem 1, the optimal decision for \mathbf{s} must be $L_a^* - 1 \neq L^*$ which is a contradiction. Moreover, if $L_a^* = 0$, from the second point of Theorem 1, \mathbf{s} must be a non-offloading state, leading to another contradiction. This proves the third point of Theorem 1.

B. Proof of Theorem 2

We recall that given an current state \mathbf{s} , $\bar{\mathbf{s}}_{1L}$ denotes the resulting state by offloading L most imminent task from \mathbf{s} . If $L^* = 0$ is the optimal offloading decision of state \mathbf{s} , then, \mathbf{s} is a non-offloading state. It is trivially that $L^* = 0$ is the smallest offloading decision to reach a non-offloading state.

Let us consider the sequence of states $\bar{\mathbf{s}}_{10} = \mathbf{s}, \bar{\mathbf{s}}_{11}, \bar{\mathbf{s}}_{12}, \dots, \bar{\mathbf{s}}_{1i}, \dots$. By definition of the notation $\bar{\mathbf{s}}_{1i}$, state $\bar{\mathbf{s}}_{1(i+1)}$ is obtained by offloading the most imminent task from state $\bar{\mathbf{s}}_{1i}$ in the sequence. Therefore, state $\bar{\mathbf{s}}_{1i}$ is adjacent to $\bar{\mathbf{s}}_{1(i+1)}$. Assume $L^* > 0$ is the optimal offloading decision for \mathbf{s} . From the first point of Theorem 1, the optimal offloading decision of state $\bar{\mathbf{s}}_{11}$ would be $L_1^* = L^* - 1$. By alternatively applying this property, the optimal offloading decisions L_i^* of states $\bar{\mathbf{s}}_{1i}$ for $i = 1, \dots, L^*$ can be derived as

$$L_i^* = L^* - i, \text{ for } i = 1, \dots, L^*. \quad (56)$$

The above result suggests that the optimal offloading decision of the state $\bar{\mathbf{s}}_{1L^*}$ would be $L_i^* = 0$ for $i = L^*$. From the second point of Theorem 1, the optimal offloading decision of $\bar{\mathbf{s}}_{1(L^*+1)}$ would also be 0. Repeatedly applying this property allows us to derive the optimal decisions for state $\bar{\mathbf{s}}_{1i}, i > L^*$ as follows:

$$L_i^* = 0, \text{ for } i > L^*. \quad (57)$$

In conclusion, states $\bar{\mathbf{s}}_{1i}$ for $i \leq L^* - 1$ are offloading states, and states $\bar{\mathbf{s}}_{1i}$ for $i \geq L^*$ are non-offloading states. Therefore, $L^* > 0$ is the smallest offloading decision to reach a non-offloading state $\bar{\mathbf{s}}_{1L^*}$.

X. APPENDIX B: PROOFS OF LEMMAS

A. Proof of Lemma 1

A sequence (a_1, a_2, \dots, a_N) of non-negative integers is called *Catalan* if

$$1 \leq a_1 \leq a_2 \leq \dots \leq a_N \text{ and } a_i \leq i, \text{ for all } 1 \leq i \leq N. \quad (58)$$

In the proof we show that there is a one-to-one correspondence between reduced sequences and Catalan sequences of the same length N . The correspondence is defined as follows.

Given a reduced sequence (n_1, n_2, \dots, n_N) , which satisfies inequalities (22), define a sequence (a_1, a_2, \dots, a_N) as follows

$$a_i = (n_1 + 1) + n_2 + \dots + n_i.$$

It is clear that the resulting sequence satisfies the Catalan sequence property (58).

Conversely, given a Catalan sequence (a_1, a_2, \dots, a_N) , which satisfies property (58), define the sequence (n_1, n_2, \dots, n_N) as follows

$$n_i = \begin{cases} 0 & \text{if } i = 1 \\ a_i - a_{i-1} & \text{if } 2 \leq i \leq N \end{cases}$$

The resulting sequence contains elements of a reduced state vector because

$$\begin{aligned} n_1 + n_2 + n_3 + \cdots + n_i &= 0 + a_2 - a_1 + a_3 - a_2 + \cdots + a_i - a_{i-1} \\ &= a_i - a_1 \\ &\leq i - 1, \end{aligned}$$

since $a_1 = 1$. Also observe that the resulting correspondence between reduced and Catalan sequences of the same length N is one-to-one.

The proof of this lemma is now complete since in exercise 78 from [11], the number of Catalan sequences of length N is equal to the Catalan number C_N .

B. Proof of Lemma 2

We assume that an offloading state $\mathbf{s} = (n_1, \dots, n_N)$ is given with d is the deadline of the most imminent task. It is trivially that when $\mathbf{s} = (0, \dots, 0)$, the optimal decision is offloading 0 task. When $N \geq 2$, in the following two cases, the optimal decision is offloading the most imminent task:

- $d \leq N - 1$, and $n_i = 0$ for $i = d + 1, \dots, N$.
- $d = N$.

Examples for these cases are given below:

Example 5: In the following two examples, it is trivially that the optimal policy offloads the most imminent tasks:

- $\mathbf{s} = (0, 0, 4, 0, 0)$ where $N = 5$, $d = 3$, and $n_i = 0, i = 4, 5$.
- $\mathbf{s} = (0, 0, 0, 0, 7)$ where $d = N = 5$.

Now, we will consider the remaining case which is: $N \geq 2$, and $d \leq N - 1$, and there exists a deadline $d' \in \{d + 1, \dots, N\}$ such that $n_{d'} > 0$. This indicates that $n_i = 0, i = 1, \dots, d - 1$, $n_d \geq 1$, and $n_{d'} \geq 1$. Let us recall the following notation:

- $\bar{\mathbf{s}}_{d1}$ is the state obtained by offloading a task at deadline d from state \mathbf{s} .
- $\bar{\mathbf{s}}_{d'1}$ is the state obtained by offloading a task at deadline d' from state \mathbf{s} .

We notice that the i^{th} elements of state $\bar{\mathbf{s}}_{d1}$ is defined by

$$\begin{cases} n_d - 1, & \text{if } i = d, \\ n_i, & \text{if } i \neq d. \end{cases} \quad (59)$$

The i^{th} elements of state $\bar{\mathbf{s}}_{d'1}$ is defined by

$$\begin{cases} n_{d'} - 1, & \text{if } i = d', \\ n_i, & \text{if } i \neq d'. \end{cases} \quad (60)$$

Therefore, we have: $\bar{\mathbf{s}}_{d1} \in \mathbb{S}_{pp}(\bar{\mathbf{s}}_{d'1})$. From Proposition 2, we have

$$J_T(\bar{\mathbf{s}}_{d1}) \leq J_T(\bar{\mathbf{s}}_{d'1}), \text{ for every given time horizon } T. \quad (61)$$

This indicates that state $\bar{\mathbf{s}}_{d1}$ is associated with a lower average cost than that of state $\bar{\mathbf{s}}_{d'1}$ for every deadline $d' \in \{d+1, \dots, N\}$. Therefore, if \mathbf{s} is an offloading state, it is optimal to offload the most imminent task from \mathbf{s} . Moreover, from the above proof, we can also conclude that whenever the local processing is available, it is optimal to process the most imminent task in \mathbf{s} .

To this end, by considering state $\bar{\mathbf{s}}_{d1}$ in the place of state \mathbf{s} and repeating the proof above, we have that: If $\bar{\mathbf{s}}_{d1}$ is an offloading state, it is optimal to offload the most imminent task from $\bar{\mathbf{s}}_{d1}$. Therefore, we can conclude that: If \mathbf{s} is an offloading state and the corresponding optimal offloading decision is 2, it is optimal to offload 2 most imminent tasks from \mathbf{s} . Keeping repeating the same analysis leads us to the final conclusion as follows: If \mathbf{s} is an offloading state, and the corresponding optimal offloading decision is L^* , it is optimal to offload the L^* most imminent tasks from \mathbf{s} .

C. Proof of Lemma 3

From the definition of function $F^{\bar{\mathbf{A}}}(T, \mathbf{s}, d, L)$ in Eq. (28), we notice that LC_o is discrete linear, and hence, discrete convex with respect to L . Therefore, in order to prove that $F^{\bar{\mathbf{A}}}(T, \mathbf{s}, d, L)$ is discrete convex with respect to L , we need to prove that

$$f^{\bar{\mathbf{A}}}(T, \mathbf{s}, d, L) = J_T^{\bar{\mathbf{A}}}(\bar{\mathbf{s}}_{dL}) \quad (62)$$

is discrete convex with respect to L . To begin with, let us define two other functions as follows:

$$f(T, \mathbf{s}, d, L) = J_T(\bar{\mathbf{s}}_{dL}), \quad (63)$$

$$f^{\mathbf{A}}(T, \mathbf{s}, d, L) = J_T^{\mathbf{A}}(\bar{\mathbf{s}}_{dL}). \quad (64)$$

The three functions $f(T, \mathbf{s}, d, L)$, $f^A(T, \mathbf{s}, d, L)$, and $f^{\bar{A}}(T, \mathbf{s}, d, L)$ are characterized by the parameters T , \mathbf{s} , and d . All of them take L as variable. The domains of these three functions are the same as that of $F^{\bar{A}}(T, \mathbf{s}, d, L)$ which are given in Eqs. (29) and (30).

The definitions of $J_T^A(\cdot)$ and $J_T^{\bar{A}}(\cdot)$ are given in Eqs. (19) and (20). In this subsection, we re-express $J_T^A(\bar{\mathbf{s}}_{dL})$ and $J_T^{\bar{A}}(\bar{\mathbf{s}}_{dL})$ as follows:

$$J_T^A(\bar{\mathbf{s}}_{dL}) = \min_{L' \in \mathbb{L}(\bar{\mathbf{s}}_{dL})} \left\{ \mathcal{C}^A(\bar{\mathbf{s}}_{dL}, L') + G^A(\bar{\mathbf{s}}_{dL}, L') \right\}, \quad (65)$$

$$J_T^{\bar{A}}(\bar{\mathbf{s}}_{dL}) = \mathcal{C}^{\bar{A}}(\bar{\mathbf{s}}_{dL}) + G^{\bar{A}}(\bar{\mathbf{s}}_{dL}) = \mathcal{C}^A(\bar{\mathbf{s}}_{dL}, 0) + G^A(\bar{\mathbf{s}}_{dL}, 0). \quad (66)$$

By denoting $M_d = \sum_{i=d}^N n_i$. We can express the functions $f^A(T, \mathbf{s}, d, L)$ and $f^{\bar{A}}(T, \mathbf{s}, d, L)$ as follows:

$$f^A(T, \mathbf{s}, d, L) = \min_{L \in \mathbb{L}_d(\mathbf{s})} \min_{\hat{L} \in \{0, \dots, M_d - L\}} \left\{ (L + \hat{L}) C_o + G_{T-1}(\mathbf{s}, d, L + \hat{L}) \right\} + \mathbf{1}_{d \geq 2} n_1 C_p, \quad (67)$$

$$f^{\bar{A}}(T, \mathbf{s}, d, L) = G_{T-1}(\mathbf{s}, d, L) + \mathbf{1}_{d \geq 2} n_1 C_p. \quad (68)$$

In (67), we denote $\bar{L} = L + \hat{L}$. Then, the average future cost function $G_T(\mathbf{s}, d, \bar{L} = L + \hat{L})$ is given by

$$G_{T-1}(\mathbf{s}, d, \bar{L}) = \mu \sum_{k=0}^N p_k J_{T-1}(\mathbf{s}'_{d\bar{L}k}) + (1 - \mu) \sum_{k=0}^N p_k J_{T-1}(\mathbf{s}''_{d\bar{L}k}) \quad (69)$$

where $\mathbf{s}'_{d\bar{L}k}$ is the state transited from \mathbf{s} with the following steps: offloading $\bar{L} = L + \hat{L}$ most imminent tasks starting from deadline d , deadline shifting, a task arrives with deadline k ($k = 0$ implies no task arrival), and local processing. $\mathbf{s}''_{d\bar{L}k}$ is defined similarly but without the local processing at the end.

Subsequently, we will show that each term $J_{T-1}(\cdot)$ on the right-hand side of Eq. (69) can be presented by the family of function f . Let us consider an initial state $\mathbf{s} = (n_1, \dots, n_N) \neq (0, \dots, 0)$. First of all, we need to find states \mathbf{s}^1 and \mathbf{s}^2 such that $\mathbf{s}^1_{d\bar{L}} \equiv \mathbf{s}'_{d\bar{L}k}$ and $\mathbf{s}^2_{d\bar{L}} \equiv \mathbf{s}''_{d\bar{L}k}$ where $\mathbf{s}^1_{d\bar{L}}$ and $\mathbf{s}^2_{d\bar{L}}$ are obtained by removing \bar{L} most imminent tasks having deadline greater than or equal to d from \mathbf{s}^1 and \mathbf{s}^2 , respectively. Then, we will have the followings:

$$f(T-1, \mathbf{s}^1, d', L') = J_{T-1}(\bar{\mathbf{s}}^1_{d'L'}) = J_{T-1}(\mathbf{s}'_{d\bar{L}k}), \quad (70)$$

$$f(T-1, \mathbf{s}^2, d'', L'') = J_{T-1}(\bar{\mathbf{s}}^2_{d''L''}) = J_{T-1}(\mathbf{s}''_{d\bar{L}k}). \quad (71)$$

Hereafter, we will defined \mathbf{s}^1 , \mathbf{s}^2 , d' , d'' , L' , and L'' in different cases, and provide examples to support interpretation.

- When $k + 1 < d$ which implies $d \geq 2$, we have: $\mathbf{s}^1 \equiv \mathbf{s}'_k$, $\mathbf{s}^2 \equiv \mathbf{s}''_k$, $d' = d'' = d - 1$, and $L' = L'' = \bar{L}$. The state \mathbf{s}'_k is obtained from \mathbf{s} by: performing deadline shifting, adding a new task with deadline k ($k = 0$ implies no new task is added), and removing the most imminent task. State \mathbf{s}''_k is defined similarly as state \mathbf{s}'_k , but without removing the most imminent task at the end.

Example 6: As an example for the case “ $k + 1 < d$ ”, we consider an initial state $\mathbf{s} = (0, 6, 0, 5, 7)$, and a time horizon $T > 5$. We assume that $d = 3$, and $\bar{L} = 7$ most imminent tasks having deadline greater than or equal to d are offloaded, and $k = 1$ implying that a new task arrives with deadline 1 in the next time slot. Then, we have

$$\mathbf{s}'_{d\bar{L}k} = \mathbf{s}'_{371} = (6, 0, 0, 5, 0), \quad (72)$$

$$\mathbf{s}''_{d\bar{L}k} = \mathbf{s}''_{371} = (7, 0, 0, 5, 0). \quad (73)$$

The states \mathbf{s}'_k and \mathbf{s}''_k in this example are

$$\mathbf{s}'_k = \mathbf{s}'_1 = (6, 0, 5, 7, 0), \quad (74)$$

$$\mathbf{s}''_k = \mathbf{s}''_1 = (7, 0, 5, 7, 0). \quad (75)$$

With $\mathbf{s}^1 \equiv \mathbf{s}'_k$, $\mathbf{s}^2 \equiv \mathbf{s}''_k$, $d' = d'' = d - 1 = 2$, and $L' = L'' = \bar{L} = 7$ as presented above, the states $\bar{\mathbf{s}}^1_{d'L'}$ and $\bar{\mathbf{s}}^2_{d''L''}$ are given by

$$\bar{\mathbf{s}}^1_{d'L'} = \mathbf{s}^1_{27} = (6, 0, 0, 5, 0), \quad (76)$$

$$\bar{\mathbf{s}}^2_{d''L''} = \mathbf{s}^2_{27} = (7, 0, 0, 5, 0). \quad (77)$$

This shows that state $\bar{\mathbf{s}}^1_{d'L'}$ in Eq. (76) is the same as state $\mathbf{s}'_{d\bar{L}k}$ in Eq. (72). Also, state $\bar{\mathbf{s}}^2_{d''L''}$ in Eq. (77) is the same as state $\mathbf{s}''_{d\bar{L}k}$ in Eq. (73). Therefore, Eqs. (70) and (71) can be achieved.

- When $k + 1 \geq d$, let $\beta_{dk} = \sum_{i=d}^{k+1} n_i$, we have:
 - If $d \geq 2$: For $\bar{L} \leq \beta_{dk}$, $\mathbf{s}^1 \equiv \mathbf{s}'_k$, $\mathbf{s}^2 \equiv \mathbf{s}''_k$, $d' = d'' = d - 1$, and $L' = L'' = \bar{L}$. For $\bar{L} > \beta_{dk}$, $\mathbf{s}^1 \equiv \mathbf{s}'_{d\beta_{dk}k}$, $\mathbf{s}^2 \equiv \mathbf{s}''_{d\beta_{dk}k}$, $d' = d'' = k + 1$, and $L' = L'' = \bar{L} - \beta_{dk}$.
 - If $d = 1$: For $\bar{L} \leq \beta_{dk}$, $\mathbf{s}^1 \equiv \mathbf{s}'_k$, $\mathbf{s}^2 \equiv \mathbf{s}''_k$, $d' = d'' = d$, and $L' = L'' = \bar{L} - n_1$. For $\bar{L} > \beta_{dk}$, $\mathbf{s}^1 \equiv \mathbf{s}'_{d\beta_{dk}k}$, $\mathbf{s}^2 \equiv \mathbf{s}''_{d\beta_{dk}k}$, $d' = d'' = k + 1$, and $L' = L'' = \bar{L} - \beta_{dk}$.

We note that state $\mathbf{s}'_{d\beta_{dk}k}$ is obtained following the steps: offloading β_{dk} most imminent tasks having deadline greater than or equal to d from the initial state \mathbf{s} , performing

deadline shifting, adding a task with deadline k , and removing the most imminent task. State $\mathbf{s}_{d\beta_{dk}k}''$ is defined in the same way but without removing the most imminent task at the end.

Example 7: Two examples for this case corresponding to $d \geq 2$ and $d = 1$ are as follows. Let us consider an initial state $\mathbf{s} = (0, 3, 0, 7, 5)$, a time horizon $T > 5$, and $\bar{L} = 9$. For the first example, we assume that $d = 2$, and $k = 2$. We have $\beta_{dk} = \beta_{22} = 3$. The states $\mathbf{s}_{d\bar{L}k}'$ and $\mathbf{s}_{d\bar{L}k}''$ are computed as

$$\mathbf{s}_{d\bar{L}k}' = \mathbf{s}_{292}' = (0, 0, 1, 5, 0), \quad (78)$$

$$\mathbf{s}_{d\bar{L}k}'' = \mathbf{s}_{292}'' = (0, 1, 1, 5, 0). \quad (79)$$

Since $L = 9 > \beta_{dk} = 3$, we compute states $\mathbf{s}_{d\beta_{dk}k}'$ and $\mathbf{s}_{d\beta_{dk}k}''$ as follows:

$$\mathbf{s}_{d\beta_{dk}k}' = \mathbf{s}_{232}' = (0, 0, 7, 5, 0), \quad (80)$$

$$\mathbf{s}_{d\beta_{dk}k}'' = \mathbf{s}_{232}'' = (0, 1, 7, 5, 0). \quad (81)$$

As presented above, we have $\mathbf{s}^1 \equiv \mathbf{s}_k'$, $\mathbf{s}^2 \equiv \mathbf{s}_k''$, $d' = d'' = k + 1 = 3$, and $L' = L'' = \bar{L} - n_d = 9 - 3 = 6$. Therefore, states $\bar{\mathbf{s}}_{d'L'}^1$ and $\bar{\mathbf{s}}_{d''L''}^2$ are computed as

$$\bar{\mathbf{s}}_{d'L'}^1 = \mathbf{s}_{19}^1 = (0, 0, 1, 5, 0), \quad (82)$$

$$\bar{\mathbf{s}}_{d''L''}^2 = \mathbf{s}_{19}^2 = (0, 1, 1, 5, 0). \quad (83)$$

Then, state $\bar{\mathbf{s}}_{d'L'}^1$ in Eq. (82) is the same as state $\mathbf{s}_{d\bar{L}k}'$ in Eq. (78). State $\bar{\mathbf{s}}_{d''L''}^2$ in Eq. (83) is the same as state $\mathbf{s}_{d\bar{L}k}''$ in Eq. (79). Therefore, we obtained Eqs. (70) and (71).

In the second example, we assume that $d = 1$ and $k = 3$. We have $\beta_{dk} = \beta_{13} = 10$. The states $\mathbf{s}_{d\bar{L}k}'$ and $\mathbf{s}_{d\bar{L}k}''$ are

$$\mathbf{s}_{d\bar{L}k}' = \mathbf{s}_{193}' = (0, 0, 0, 6, 0), \quad (84)$$

$$\mathbf{s}_{d\bar{L}k}'' = \mathbf{s}_{193}'' = (0, 0, 1, 6, 0). \quad (85)$$

Now $\bar{L} = 9 \leq \beta_{dk} = 10$, we define state \mathbf{s}_k' and \mathbf{s}_k'' as follows:

$$\mathbf{s}_k' = (2, 0, 7, 6, 0), \quad (86)$$

$$\mathbf{s}_k'' = (3, 0, 7, 6, 0). \quad (87)$$

We have $\mathbf{s}^1 \equiv \mathbf{s}'_k$, $\mathbf{s}^2 \equiv \mathbf{s}''_k$, $d' = d'' = d = 1$, and $L' = L'' = \bar{L} - n_1 = 9$. Thus, states $\bar{\mathbf{s}}^1_{d'L'}$ and $\bar{\mathbf{s}}^2_{d''L''}$ are computed as

$$\bar{\mathbf{s}}^1_{d'L'} = \mathbf{s}^1_{19} = (0, 0, 0, 6, 0), \quad (88)$$

$$\bar{\mathbf{s}}^2_{d''L''} = \mathbf{s}^2_{19} = (0, 0, 1, 6, 0)., \quad (89)$$

which suggests that state $\bar{\mathbf{s}}^1_{d'L'}$ in Eq. (88) is the same as state $\mathbf{s}'_{d\bar{L}k}$ in Eq. (84). State $\bar{\mathbf{s}}^2_{d''L''}$ in Eq. (89) is the same as state $\mathbf{s}''_{d\bar{L}k}$ in Eq. (85). As a result, we achieve Eqs. (70) and (71).

From the presented cases, the terms $J_{T-1}(\mathbf{s}'_{d\bar{L}k})$ and $J_{T-1}(\mathbf{s}''_{d\bar{L}k})$ are expressed by functions f as follows:

- If $k + 1 < d$:

$$J_{T-1}(\mathbf{s}'_{d\bar{L}k}) = f(T - 1, \mathbf{s}'_k, d - 1, \bar{L}), \quad (90)$$

$$J_{T-1}(\mathbf{s}''_{d\bar{L}k}) = f(T - 1, \mathbf{s}''_k, d - 1, \bar{L}). \quad (91)$$

- If $k + 1 \geq d$ and $d \geq 2$:

$$J_{T-1}(\mathbf{s}'_{d\bar{L}k}) = g'_{dk}(\bar{L}) = \begin{cases} f(T - 1, \mathbf{s}'_k, d - 1, \bar{L}), & \text{if } \bar{L} \leq \beta_{dk}, \\ f(T - 1, \mathbf{s}'_{d\beta_{dk}k}, k + 1, \bar{L} - \beta_{dk}), & \text{if } \bar{L} > \beta_{dk}, \end{cases} \quad (92)$$

$$J_{T-1}(\mathbf{s}''_{d\bar{L}k}) = g''_{dk}(\bar{L}) = \begin{cases} f(T - 1, \mathbf{s}''_k, d - 1, \bar{L}), & \text{if } \bar{L} \leq \beta_{dk}, \\ f(T - 1, \mathbf{s}''_{d\beta_{dk}k}, k + 1, \bar{L} - \beta_{dk}), & \text{if } \bar{L} > \beta_{dk}, \end{cases} \quad (93)$$

where $\beta_{dk} = \sum_{i=d}^{k+1} n_i$.

- If $k + 1 \geq d$ and $d = 1$:

$$J_{T-1}(\mathbf{s}'_{1\bar{L}k}) = g'_{1k}(\bar{L}) = \begin{cases} f(T - 1, \mathbf{s}'_k, 1, \bar{L}), & \text{if } \bar{L} \leq \beta_{1k}, \\ f(T - 1, \mathbf{s}'_{1\beta_{1k}k}, k + 1, \bar{L} - \beta_{1k}), & \text{if } \bar{L} > \beta_{1k}, \end{cases} \quad (94)$$

$$J_{T-1}(\mathbf{s}''_{1\bar{L}k}) = g''_{1k}(\bar{L}) = \begin{cases} f(T - 1, \mathbf{s}''_k, 1, \bar{L}), & \text{if } \bar{L} \leq \beta_{1k}, \\ f(T - 1, \mathbf{s}''_{1\beta_{1k}k}, k + 1, \bar{L} - \beta_{1k}), & \text{if } \bar{L} > \beta_{1k}, \end{cases} \quad (95)$$

where $\beta_{1k} = \sum_{i=1}^{k+1} n_i$.

In the above, $g'_{dk}(\bar{L})$ and $g''_{dk}(\bar{L})$ are functions characterized by parameters d and k , and take \bar{L} as their variables.

Next, we will prove the convexity of functions f using induction, starting with an initial case.

Initial case: We consider state $\mathbf{s} = (n_1, \dots, n_N)$, and a time horizon $T = 1$.

In this case, if $N = 1$, we have $\mathbf{s} = (n_1)$, then, there is only one valid offloading decision $\mathbb{L}_1(\mathbf{s}) = \{n_1\}$. For $N \geq 2$, it is trivially that all tasks having deadline 1 are excessive tasks, and should be offloaded whenever the AMA is available, which results in a cost $n_1 C_o$. If the AMA is not available at the initial time slot, tasks with deadline 1 will expire and result in a cost $n_1 C_p$. Since in this case, we are considering a time horizon with only one time slot, the minimum average cost can be computed straightforwardly. Thus, we have the following

$$f(1, \mathbf{s}, 1, L) = J_1(\mathbf{s}_{1L}) = (p_u C_o + (1 - p_u) C_p) n_1 + (L - n_1) C_o, \text{ for } L \in \mathbb{L}_1(\mathbf{s}), \quad (96)$$

$$f(1, \mathbf{s}, d, L) = J_1(\mathbf{s}_{dL}) = n_1 C_p + L C_o, \text{ for } d \geq 2 \text{ and } L \in \mathbb{L}_d(\mathbf{s}). \quad (97)$$

We recall that the smallest offloading decision in the set $\mathbb{L}_1(\mathbf{s})$ is n_1 . From Eqs. (96) and (97), $f(1, \mathbf{s}, d, L)$, $d = 1, \dots, N$ are discrete linear function with respect to L , hence, they are discrete convex function with respect to L . Next is an inductive step where we prove the convexity of $f(T, \mathbf{s}, d, L)$ given that of $f(T-1, \mathbf{s}, d, L)$ for every parameters \mathbf{s} and d .

Inductive step: Let us make the following assumption: *The functions $f(T-1, \mathbf{s}, d, L)$ is discrete convex with respect to L for every given state \mathbf{s} and deadline d .*

Now, we consider Eqs. (90)-(95) with \bar{L} is replaced by $L + \hat{L}$. From the above assumption, $f(T-1, \mathbf{s}'_k, d-1, L + \hat{L})$ in Eq. (90), and $f(T-1, \mathbf{s}''_k, d-1, L + \hat{L})$ in Eq. (91) are discrete convex with respect to L . Next, we will prove that $g'_{dk}(L + \hat{L})$ in Eq. (92) is discrete convex with respect with respect to L . Then, the convexity of functions $g''_{dk}(L + \hat{L})$ in Eq. (93), $g'_{1k}(L + \hat{L})$ in Eq. (94), and $g''_{1k}(L + \hat{L})$ in Eq. (95) can also be proved in a very similar way.

Let us consider function $g'_{dk}(L + \hat{L})$ in Eq. (92). With our assumption above, $f(T-1, \mathbf{s}'_k, d-1, L + \hat{L})$ for $L + \hat{L} \leq \beta_{dk}$, and $f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, L + \hat{L} - \beta_{dk})$ for $L + \hat{L} > \beta_{dk}$ are discrete convex with respect to L . Therefore, in order to prove the convexity of $g'_{dk}(L + \hat{L})$ we will prove that the discrete Jensen's inequality holds at the connecting point of $f(T-1, \mathbf{s}'_k, d-1, L + \hat{L})$ and $f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, L + \hat{L} - \beta_{dk})$, i.e., at $\bar{L} = \beta_{dk}$. To be more specific, we will prove the following:

$$f(T-1, \mathbf{s}'_k, d-1, \beta_{dk}) + f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 2) \geq 2f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 1). \quad (98)$$

From the definition of function f in Eq. (63), we have

$$f(T-1, \mathbf{s}'_k, d-1, \beta_{dk}) = J_{T-1}(\mathbf{s}'_{d\beta_{dk}k}), \quad (99)$$

and also

$$f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 0) = J_{T-1}(\mathbf{s}'_{d\beta_{dk}k}). \quad (100)$$

Hence,

$$f(T-1, \mathbf{s}'_k, d-1, \beta_{dk}) = f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 0). \quad (101)$$

Moreover, since $f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, L + \hat{L})$ is discrete convex with respect to L due to our assumption, we have

$$f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 0) + f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 2) \geq 2f(T-1, \mathbf{s}'_{d\beta_{dk}k}, k+1, 1). \quad (102)$$

By combining Eq. (101) and Ineq. (102), we prove that Ineq. (98) is true. Therefore, $g'_{dk}(L + \hat{L})$ in Eq. (92) is discrete convex with respect to L . Similarly, $g''_{dk}(L + \hat{L})$ in Eq. (93), $g'_{1k}(L + \hat{L})$ in Eq. (94), and $g''_{1k}(L + \hat{L})$ in Eq. (95) are also discrete convex with respect to L .

By using Eqs. (90)-(95), we can express $G_{T-1}(\mathbf{s}, d, L + \hat{L})$ defined in Eq. (69) (where \bar{L} is replaced by $L + \hat{L}$) by the sum of multiple functions f having $T-1$ as the first argument. Furthermore, these functions are discrete convex with respect to L according to our assumption at the beginning of this **Inductive step**. Therefore, $G_{T-1}(\mathbf{s}, d, L + \hat{L})$ is a discrete convex function with respect to L .

Now, we will refer to the constant term $\mathbb{1}_{d \geq 2} n_1 C_p$ in (67) as *constant*, and denote

$$g(L + \hat{L}) = (L + \hat{L}) C_o + G_T(\mathbf{s}, d, L + \hat{L}). \quad (103)$$

where $G_T(\mathbf{s}, d, L + \hat{L})$ has been proved to be a discrete convex function. In addition, $(L + \hat{L}) C_o$ is discrete linear, hence, also discrete convex. As a result, $g(L + \hat{L})$ is a discrete convex function with respect to L . Then, Eq. (67) becomes

$$f^A(T, \mathbf{s}, d, L) = \min_{\hat{L} \in \{0, 1, \dots, M_d - L\}} \{g(L + \hat{L})\} + \text{constant}. \quad (104)$$

Assuming that $g(L + \hat{L})$ attains its minimum $g^{\min}(L^*)$ at $L^* \geq n_1$. Note that the condition $L^* \geq n_1$ holds because all tasks having deadline 1 are excessive tasks, and will result in a

penalty C_p per task if not offloaded. Therefore, given that $C_p > C_o$, the optimal offloading decision cannot be less than n_1 . Then, we have

$$f^A(T, \mathbf{s}, d, L) = \begin{cases} g^{\min}(L^*) + \text{constant}, & \text{if } L \leq L^*, \\ g(L) + \text{constant where } \hat{L} = 0, & \text{if } L > L^*. \end{cases} \quad (105)$$

As a results, $f^A(T, \mathbf{s}, d, L)$ are discrete convex functions with respect to $L \in \mathbb{L}_d(\mathbf{s})$ for all states \mathbf{s} and deadlines d .

$G_{T-1}(\mathbf{s}, d, L)$ in Eq. (68) can be expressed in a similar fashion as Eq. (69) with \bar{L} is replaced by L , since without the AMA, $\hat{L} = 0$. Thus, $G_{T-1}(\mathbf{s}, d, L)$ in Eq. (68) can also be written as the sum of several functions f , each has $T-1$ as its first argument. This suggests that in Eq. (68), $G_{T-1}(\mathbf{s}, d, L)$ is a discrete convex function with respect to L for every given \mathbf{s} and d , and so is $f^{\bar{A}}(T, \mathbf{s}, d, L)$. Afterwards, by combining our definitions presented in Eqs. (62)-(64) with the following equation:

$$J_T(\bar{\mathbf{s}}_{dL}) = p_u J_T^A(\bar{\mathbf{s}}_{dL}) + (1 - p_u) J_T^{\bar{A}}(\bar{\mathbf{s}}_{dL}), \quad (106)$$

we have

$$f(T, \mathbf{s}, d, L) = p_u f^A(T, \mathbf{s}, d, L) + (1 - p_u) f^{\bar{A}}(T, \mathbf{s}, d, L). \quad (107)$$

Therefore, $f(T, \mathbf{s}, d, L)$ is also a discrete convex function with respect to L for every given state \mathbf{s} and deadline d .

This result together with the presented **Initial case** lead us to a conclusion that $f(T, \mathbf{s}, d, L)$ are discrete convex functions with respect to L for all parameters T , \mathbf{s} , and d . As $G_T(\mathbf{s}, d, L)$ is expressed by the sum of multiple functions f , so is $f^{\bar{A}}(T, \mathbf{s}, d, L)$, which is shown in Eq. (68). Therefore, $f^{\bar{A}}(T, \mathbf{s}, d, L)$ is also discrete convex functions with respect to L . Then, from our discussion at the beginning of this subsection, we can conclude that $F^{\bar{A}}(T, \mathbf{s}, d, L)$ are discrete convex functions with respect to L for all parameters T , \mathbf{s} , and d . This completes our proof.

D. Proof of Lemma 4

We assume that function $F^{\bar{A}}(T, \mathbf{s}, d, L)$ attains its minimum at L^* for $d = 1$, the following set of inequalities hold

$$J_T^{\bar{A}}(\bar{\mathbf{s}}_{1L^*}) + L^* C_o \leq J_T^{\bar{A}}(\bar{\mathbf{s}}_{1L}) + L C_o, \text{ for all } L \in \mathbb{L}_1(\mathbf{s}). \quad (108)$$

From Eq. (40) and Ineqs. (108), we have

$$J_T^A(\mathbf{s}, L^*) \leq J_T^A(\mathbf{s}, L), \text{ for all } L \in \mathbb{L}_1(\mathbf{s}). \quad (109)$$

Equivalently,

$$J_T^A(\mathbf{s}, L^*) = \min_{L \in \mathbb{L}_1(\mathbf{s})} \{J_T^A(\mathbf{s}, L)\}. \quad (110)$$

We recall a fact that the optimal offloading decision must not be less than the number of tasks having deadline 1 which are all excessive tasks. Thus, L^* is the optimal offloading decision of \mathbf{s} for the time horizon T .

XI. APPENDIX C: PROOFS OF PROPOSITIONS

A. Proof of Proposition 1

Considering a state $\mathbf{s} = (n_1, \dots, n_N)$ and a time horizon T . Let $\mathbf{s}_r = (n_1^r, \dots, n_N^r)$ be the corresponding reduced state obtained via Algorithm IV.1, and let $\mathbf{s}_m = (n_1^m, \dots, n_N^m)$ denote the corresponding lean state obtained according to Definition 3. As n_i^m , $i = 1, \dots, N$, are defined by Eq. (24) with the parameters γ_i , $i = 1, \dots, N$, are given in Eq. (23), we have

$$n_i^r \leq n_i^m \leq n_i, \quad i = 1, \dots, N. \quad (111)$$

Moreover, we recall that $n_i - n_i^r$ tasks having deadline i in \mathbf{s} , $i = 1, \dots, N$ are excessive tasks, i.e., they are guaranteed to expire if not offloaded within the first i time slots. Therefore, $n_i - n_i^m$ tasks having deadline i in \mathbf{s} , $i = 1, \dots, N$ are also excessive tasks. In other words, for each deadline, tasks that \mathbf{s} has more than \mathbf{s}_m are excessive tasks.

Example 8: For example, with $\mathbf{s} = (0, 3, 4, 0, 5)$, the corresponding lean state would be $\mathbf{s}_r = (0, 1, 1, 0, 4)$. Then, for deadline 1, there are $3 - 1 = 2$ excessive tasks. For deadline 2, there are $4 - 1 = 3$ excessive tasks. For deadline 4 and 5, there is 0 excessive task.

Now let us consider the following cases in which we use the notations $J_T(\mathbf{s})|_j$ and $J_T(\mathbf{s}_m)|_j$ to denote the minimum average cost of \mathbf{s} and \mathbf{s}_m , respectively, given that the AMA arrives for the first time at time slot t . Then, there are following cases.

- Case 0: The AMA is available for the first time at the current time slot, $t = 0$. As we mentioned, for every deadline, tasks that state \mathbf{s} has more than state \mathbf{s}_m are excessive tasks which should be offloaded by the optimal policy whenever the AMA is available. Therefore, in this case, if L_m^* denotes the optimal number of tasks to offload of \mathbf{s}_m , that

of \mathbf{s} will be $L_m^* + y$, where $y = \sum_{i=1}^N (n_i - n_i^m)$ is the partial number of excessive tasks in \mathbf{s} . Hence,

$$J_T(\mathbf{s}) \big|_0 = J_T(\mathbf{s}_m) \big|_0 + C_o \sum_{i=1}^N (n_i - n_i^m). \quad (112)$$

- Case 1: If the AMA is available for the first time at time slot 1, the number of tasks having deadline 1 expiring from \mathbf{s} is more than that from \mathbf{s}_m by $n_1 - n_1^m$. All the other remaining excessive tasks can be offloaded from both states \mathbf{s} and \mathbf{s}_m . Hence,

$$J_T(\mathbf{s}) \big|_1 = J_T(\mathbf{s}_m) \big|_1 + C_o \sum_{i=2}^N (n_i - n_i^m) + (n_1 - n_1^m) C_p. \quad (113)$$

The same logic can be applied to other cases when the AMA first arrives at time slot $2, 3, \dots, N$. Therefore, we present next the last case.

- Case N : If the AMA is available for the first time at time slot N , the number of tasks having deadline i expiring from \mathbf{s} is more than that from \mathbf{s}_m by $n_i - n_i^m$. Therefore, we have

$$J_T(\mathbf{s}) \big|_N = J_T(\mathbf{s}_m) \big|_N + C_p \sum_{i=1}^N (n_i - n_i^m). \quad (114)$$

The probability that the AMA arrives for the first time at time slot t is computed as

$$P_t = p_u (1 - p_u)^t. \quad (115)$$

Also, the probability that the AMA does not arrive within the first N time slots is

$$P_{t \geq N} = (1 - p_u)^N. \quad (116)$$

From the above logic, $J_T(\mathbf{s})$ can be expressed in terms of $J_T(\mathbf{s}_m)$ as follows:

$$\begin{aligned} J_T(\mathbf{s}) &= P_{t=0} \left(J_T(\mathbf{s}_m) \big|_0 + C_o \sum_{i=1}^N (n_i - n_i^m) \right) \\ &+ P_{t=1} \left(J_T(\mathbf{s}_m) \big|_1 + C_o \sum_{i=2}^N (n_i - n_i^m) + C_p (n_1 - n_1^m) \right) \\ &+ \dots \\ &+ P_{t=N-1} \left(J_T(\mathbf{s}_m) \big|_{N-1} + C_o (n_N - n_N^m) + C_p \sum_{i=1}^{N-1} (n_i - n_i^m) \right) \\ &+ P_{t \geq N} \left(J_T(\mathbf{s}_m) \big|_{t \geq N} + C_p \sum_{i=1}^N (n_i - n_i^m) \right). \end{aligned} \quad (117)$$

$J_T(\mathbf{s}_m)$ is the minimum cost averaged over all cases, hence, can be expressed by

$$J_T(\mathbf{s}_m) = \sum_{i=0}^{N-1} P_{t=i} J_T(\mathbf{s}_m) \big|_i + P_{t \geq N} J_T(\mathbf{s}_m) \big|_{t \geq N}. \quad (118)$$

In conclusion, with the aid of Eq. (118), the equation (117) can be simplified to Eq. (25).

B. Proof of Proposition 2

Given an initial state $\mathbf{s} = (n_1, \dots, n_N) \neq (0, \dots, 0)$, and a time horizon T . We call d the deadline of the most imminent task of \mathbf{s} , i.e., d is the smallest deadline such that $n_d > 0$. Denoting $\tilde{\mathbf{s}} = (\tilde{n}_1, \dots, \tilde{n}_N) \in \mathbb{S}_{pp}(\mathbf{s})$. Trivially, we have

$$J_T(\tilde{\mathbf{s}}) = J_T(\mathbf{s}), \text{ if } N = 1. \quad (119)$$

If $N \geq 2$, by definition of set $\mathbb{S}_{pp}(\mathbf{s})$, there must be a deadline $\tilde{d} \in \{d, \dots, N\}$ such that

$$\tilde{n}_i = \begin{cases} n_i, & \text{if } i \neq \tilde{d} \text{ and } i \neq d, \\ n_d - 1, & \text{if } i = d, \\ n_{\tilde{d}} + 1, & \text{if } i = \tilde{d}. \end{cases} \quad (120)$$

If $\tilde{d} = d$, $\mathbf{s} \equiv \tilde{\mathbf{s}}$. It is trivially that

$$J_T(\tilde{\mathbf{s}}) = J_T(\mathbf{s}), \text{ if } N \geq 2 \text{ and } \tilde{d} = d. \quad (121)$$

For the rest of this part, we will prove that

$$J_T(\tilde{\mathbf{s}}) \leq J_T(\mathbf{s}), \text{ when } \tilde{d} \geq d + 1, d \leq N - 1, N \geq 2. \quad (122)$$

We conduct our proof via induction starting with an initial case.

Initial case: Given $\mathbf{s} = (n_1, \dots, n_N)$, $\tilde{\mathbf{s}} = (\tilde{n}_1, \dots, \tilde{n}_N) \in \mathbb{S}_{pp}(\mathbf{s})$, and $T = 1$.

- If $n_1 = 0$, $\tilde{n}_1 = 0$ according to Eq. (120). Therefore,

$$J_1(\tilde{\mathbf{s}}) = J_1(\mathbf{s}) = 0. \quad (123)$$

- If $n_1 \geq 1$, $\tilde{n}_1 = n_1 - 1$ according to Eq. (120). We have

$$J_1(\mathbf{s}) = (p_u C_o + (1 - p_u) C_p) n_1, \quad (124)$$

$$J_1(\tilde{\mathbf{s}}) = (p_u C_o + (1 - p_u) C_p) (n_1 - 1). \quad (125)$$

Therefore,

$$J_1(\tilde{\mathbf{s}}) \leq J_1(\mathbf{s}). \quad (126)$$

The subsequent part is an inductive step.

Inductive step: In this step, we make the following assumption: *For every given state \mathbf{s} and $\tilde{\mathbf{s}} \in \mathbb{S}_{pp}(\mathbf{s})$, the following holds*

$$J_{T-1}(\tilde{\mathbf{s}}) \leq J_{T-1}(\mathbf{s}), \quad (127)$$

Initially, we highlight an observation in the following remark.

Remark 1: Given two states \mathbf{s} and $\tilde{\mathbf{s}}$ which are assumed to transit to \mathbf{s}' and $\tilde{\mathbf{s}}'$ in the next time slot, respectively, if no offloading is performed. If $\tilde{\mathbf{s}} \in \mathbb{S}_{pp}(\mathbf{s})$ and the number of tasks having deadline 1 of \mathbf{s} is 0, then, $\tilde{\mathbf{s}}' \in \mathbb{S}_{pp}(\mathbf{s}')$.

Remark 1 can be proved as follows. Given state \mathbf{s} and $\tilde{\mathbf{s}} \in \mathbb{S}_{pp}(\mathbf{s})$ as above with $n_1 = 0$. We assume that no offloading is performed, and that \mathbf{s} and $\tilde{\mathbf{s}}$ becomes $\mathbf{s}_{ds} = (n_1^{ds}, \dots, n_N^{ds})$ and $\tilde{\mathbf{s}}_{ds} = (\tilde{n}_2^{ds}, \dots, \tilde{n}_N^{ds})$, respectively, after the deadline shifting, in which

$$n_i^{ds} = \begin{cases} n_{i+1}, & \text{if } i \neq N, \\ 0, & \text{if } i = N, \end{cases} \quad (128)$$

and

$$\tilde{n}_i^{ds} = \begin{cases} n_i^{ds}, & \text{if } i \neq \tilde{d} - 1 \text{ and } i \neq d - 1, \\ n_d^{ds} - 1, & \text{if } i = d - 1, \\ n_d^{ds} + 1, & \text{if } i = \tilde{d} - 1, \\ 0, & \text{if } i = N. \end{cases} \quad (129)$$

Eqs. (128) and (129) indicate that $\tilde{\mathbf{s}}_{ds} \in \mathbb{S}_{pp}(\mathbf{s}_{ds})$. Therefore, with the same realizations of task arrival and local processing, \mathbf{s}_{ds} and $\tilde{\mathbf{s}}_{ds}$ would transit to states \mathbf{s}' and $\tilde{\mathbf{s}}'$, respectively, where $\tilde{\mathbf{s}}' \in \mathbb{S}_{pp}(\mathbf{s}')$.

Based on the emphasized observation above, let us consider the following two cases:

Case 1: \mathbf{s} is a non-offloading state.

Case 2: The AMA does not presents at the initial time slot, and $n_1 = 0$.

We note that in **Case 1**, \mathbf{s} is a non-offloading state, implying that $n_1 = 0$. This is because tasks having deadline 1 are all excessive tasks and must be offloaded whenever the AMA is available. Applying the DP equation (14) in Case 1, we have

$$J_T(\mathbf{s}) = \mu \sum_{k=0}^N p_k J_{T-1}(\mathbf{s}'_{0k}) + (1 - \mu) \sum_{k=0}^N p_k J_{T-1}(\mathbf{s}''_{0k}), \quad (130)$$

$$J_T(\tilde{\mathbf{s}}) = \mu \sum_{k=0}^N p_k J_{T-1}(\tilde{\mathbf{s}}'_{0k}) + (1 - \mu) \sum_{k=0}^N p_k J_{T-1}(\tilde{\mathbf{s}}''_{0k}). \quad (131)$$

Applying Eq. (20) to **Case 2** yields

$$J_T^{\bar{A}}(\mathbf{s}) = \mu \sum_{k=0}^N p_k J_{T-1}(\mathbf{s}'_{0k}) + (1 - \mu) \sum_{k=0}^N p_k J_{T-1}(\mathbf{s}''_{0k}), \quad (132)$$

$$J_T^{\bar{A}}(\tilde{\mathbf{s}}) = \mu \sum_{k=0}^N p_k J_{T-1}(\tilde{\mathbf{s}}'_{0k}) + (1 - \mu) \sum_{k=0}^N p_k J_{T-1}(\tilde{\mathbf{s}}''_{0k}). \quad (133)$$

As both **Case 1** and **Case 2** match the condition mentioned in Remark 1 that $\tilde{\mathbf{s}} \in \mathbb{S}_{pp}(\mathbf{s})$ and $n_1 = 0$, we have: $\tilde{\mathbf{s}}'_{0k} \in \mathbb{S}_{pp}(\mathbf{s}'_{0k})$ and $\tilde{\mathbf{s}}''_{0k} \in \mathbb{S}_{pp}(\mathbf{s}''_{0k})$. Combining this with our assumption in Eq. (127), we have

$$J_{T-1}(\tilde{\mathbf{s}}'_{0k}) \leq J_{T-1}(\mathbf{s}'_{0k}), \quad (134)$$

$$J_{T-1}(\tilde{\mathbf{s}}''_{0k}) \leq J_{T-1}(\mathbf{s}''_{0k}). \quad (135)$$

Combining Ineqs. (134)-(135) with Eqs. (130)-(131) gives

$$J_T(\tilde{\mathbf{s}}) \leq J_T(\mathbf{s}), \text{ when } \mathbf{s} \text{ is a non-offloading state.} \quad (136)$$

Combining Ineqs. (134)-(135) with Eqs. (132)-(133) gives

$$J_T^{\bar{A}}(\tilde{\mathbf{s}}) \leq J_T^{\bar{A}}(\mathbf{s}), \text{ when } n_1 = 0. \quad (137)$$

The other case without AMA's presence is as follows:

Case 3: The AMA does not presents at the initial time slot, and $n_1 \geq 1$.

In this case, assuming that \mathbf{s} and $\tilde{\mathbf{s}}$ transit to $\mathbf{s}' = (n'_1, \dots, n'_N)$ and $\tilde{\mathbf{s}}' = (\tilde{n}'_1, \dots, \tilde{n}'_N)$ in the next time slot, respectively. Based on Eq. (120), we have

$$\tilde{n}'_i = \begin{cases} n'_i, & \text{if } i \neq \tilde{d} - 1, \\ n'_i + 1, & \text{if } i = \tilde{d} - 1. \end{cases} \quad (138)$$

We note that all tasks having deadline 1 cannot be processed and will expire, since the deadline shifting happens before the local processing event as presented in Fig. 2. Therefore, from Eq. (120), there is a fact that after the first time slot, the number of tasks expiring from \mathbf{s} is more than that from $\tilde{\mathbf{s}}$ by 1. This observation leads to the following equation.

$$J_T^{\bar{A}}(\mathbf{s}) - J_T^{\bar{A}}(\tilde{\mathbf{s}}) = C_p + J_T^{\bar{A}}(\mathbf{s}') - J_T^{\bar{A}}(\tilde{\mathbf{s}}'). \quad (139)$$

From Eq. (138), we can also observe that $\tilde{\mathbf{s}}'$ only differs from \mathbf{s}' by an additional task having deadline \tilde{d} . Furthermore, C_p is the largest cost that can possibly be incurred by a task. We have the following inequality.

$$C_p + J_T^{\bar{A}}(\mathbf{s}') - J_T^{\bar{A}}(\tilde{\mathbf{s}}') \geq 0. \quad (140)$$

Applying Ineq. (140) to Eq. (139) gives us

$$J_T^{\bar{A}}(\tilde{\mathbf{s}}) \leq J_T^{\bar{A}}(\mathbf{s}) \text{ when } n_1 \geq 1. \quad (141)$$

From Ineqs. (137) and (141), we have

$$J_T^{\bar{A}}(\tilde{\mathbf{s}}) \leq J_T^{\bar{A}}(\mathbf{s}). \quad (142)$$

Finally, it remains to consider the following case.

Case 4: The AMA presents at the initial time slot, and \mathbf{s} is an offloading state.

In this case, we consider two contexts in which both are associated with a time horizon T . In the first context, state \mathbf{s} is given initially, and L^* is the corresponding optimal offloading decision. In the second one, $\tilde{\mathbf{s}}$ is given as an initial state, and task offloading is done according to a policy $\pi(L^*)$ in the first time slot. The task offloading decision of policy $\pi(L^*)$ in the first time slot is defined as follows.

- Situation 1: $L^* \geq 1 + \sum_{i=1}^{\tilde{d}-1} n_i$, then, $\pi(L^*)$ offloads L^* most imminent tasks from $\tilde{\mathbf{s}}$.
- Situation 2: $0 < L^* < 1 + \sum_{i=1}^{\tilde{d}-1} n_i$, then, $\pi(L^*)$ offloads $L^* - 1$ most imminent tasks, and offloads a task with deadline \tilde{d} from $\tilde{\mathbf{s}}$.

For the two situations mentioned above, the policy $\pi(L^*)$ guarantees that in both contexts, we arrive at the same state after the offloading step. For intuition, we consider the following example which corresponds to Situation 1.

Example 9: Given $\mathbf{s} = (0, 0, 5, 4, 5)$ and $\tilde{\mathbf{s}} = (0, 0, 4, 4, 6) \in \mathbb{S}_{pp}(\mathbf{s})$ where $\tilde{d} = 5$. Assuming that 12 most imminent tasks are removed from \mathbf{s} resulting in state $(0, 0, 0, 0, 2)$. Since $12 > 1 + 5 + 4$, we remove 12 most imminent tasks from $\tilde{\mathbf{s}}$, which also results in state $(0, 0, 0, 0, 2)$. The next example corresponding to Situation 2 is as follows.

Example 10: Given $\mathbf{s} = (0, 0, 5, 1, 1)$ and $\tilde{\mathbf{s}} = (0, 0, 4, 1, 2) \in \mathbb{S}_{pp}(\mathbf{s})$ where $\tilde{d} = 5$. Assuming that 5 most imminent tasks are removed from \mathbf{s} resulting in state $(0, 0, 0, 1, 1)$. Since $0 < 5 < 1 + 5 + 1$, we remove $5 - 1 = 4$ most imminent tasks, and a task with deadline $\tilde{d} = 5$ from $\tilde{\mathbf{s}}$, which also gives state $(0, 0, 0, 1, 1)$.

With the AMA's availability, we denote $J_{T,\pi}^{\bar{A}}(\tilde{\mathbf{s}})$ the minimum average cost associated with state $\tilde{\mathbf{s}}$ and time horizon T given that the task offloading is done according to policy $\pi(L^*)$ at the initial time slot. Since in both contexts described above, we arrive at the same state

after the task offloading step by offloading the same number of tasks, we have the following equality

$$J_{T,\pi}^A(\tilde{\mathbf{s}}) = J_T^A(\mathbf{s}). \quad (143)$$

Trivially, $J_T^A(\tilde{\mathbf{s}}) \leq J_{T,\pi}^A(\tilde{\mathbf{s}})$. Thus, we have

$$J_T^A(\tilde{\mathbf{s}}) \leq J_T^A(\mathbf{s}). \quad (144)$$

We recall the following expression:

$$J_T(\mathbf{s}) = p_u J_T^A(\mathbf{s}) + (1 - p_u) \bar{J}_T^A(\mathbf{s}) \text{ for every state } \mathbf{s}. \quad (145)$$

Combining Ineqs. (142) and (144) with (145) gives us

$$J_T(\tilde{\mathbf{s}}) \leq J_T(\mathbf{s}), \text{ when } \mathbf{s} \text{ is an offloading state.} \quad (146)$$

Ineqs. (136) and (146) lead us to the following inequality

$$J_T(\tilde{\mathbf{s}}) \leq J_T(\mathbf{s}). \quad (147)$$

Combining Ineq. (147) with the presented **Initial case** allows us to conclude that

$$J_T(\tilde{\mathbf{s}}) \leq J_T(\mathbf{s}), \text{ for every state } \mathbf{s} \text{ and } \tilde{\mathbf{s}} \in \mathbb{S}_{pp}(\mathbf{s}). \quad (148)$$

C. Proof of Proposition 3

Given a time horizon T , if \mathbf{s}_a is a non-offloading state, the corresponding optimal offloading decision is 0, thus,

$$J_T^A(\mathbf{s}_a, 0) < J_T^A(\mathbf{s}_a, 1). \quad (149)$$

From Eq. (40), the above inequality is equivalent to

$$\bar{J}_T^A(\mathbf{s}_a) < \bar{J}_T^A(\mathbf{s}) + C_o. \quad (150)$$

Here, we have $\mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$. According to Theorem 1, if \mathbf{s}_a is a non-offloading state, \mathbf{s} is also a non-offloading state. Hence, trivially, we have

$$J_T^A(\mathbf{s}_a) = \bar{J}_T^A(\mathbf{s}_a), \quad (151)$$

$$J_T^A(\mathbf{s}) = \bar{J}_T^A(\mathbf{s}). \quad (152)$$

Combining Eqs. (151) and (152) with Eq. (18) and Ineq. (150), we have

$$J_T(\mathbf{s}_a) - J_T(\mathbf{s}) < C_o. \quad (153)$$

We make a conclusion at this point that: For two states \mathbf{s} and $\mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$, if \mathbf{s}_a is a non-offloading state, Ineq. (153) holds.

Now, we will prove the reverse in which there are two states \mathbf{s} and $\mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$, and the inequality (153) holds. We assume, in contradict, that \mathbf{s}_a is an offloading state. By applying Eq. (18) to Ineq. (153), we have

$$p_u \left(J_T^A(\mathbf{s}_a) - J_T^A(\mathbf{s}) \right) + (1 - p_u) \left(J_T^{\bar{A}}(\mathbf{s}_a) - J_T^{\bar{A}}(\mathbf{s}) \right) < C_o. \quad (154)$$

We denote $L_a^* \geq 1$ the optimal offloading decision of \mathbf{s}_a . From Theorem 1, the optimal offloading decision of \mathbf{s} would be $L_a^* - 1$. Combining this with Eq. (38), we have

$$J_T^A(\mathbf{s}_a) = J_T^A(\mathbf{s}_a, L_a^*) = J_T^A(\bar{\mathbf{s}}_{a1L_a^*}, 0) + L_a^* C_o, \quad (155)$$

$$J_T^A(\mathbf{s}) = J_T^A(\mathbf{s}, L_a^* - 1) = J_T^A(\bar{\mathbf{s}}_{1(L_a^*-1)}, 0) + (L_a^* - 1) C_o, \quad (156)$$

where state $\bar{\mathbf{s}}_{a1L_a^*}$ is obtained by offloading L_a^* most imminent tasks from \mathbf{s}_a , and $\bar{\mathbf{s}}_{1(L_a^*-1)}$ is obtained by offloading $L_a^* - 1$ most imminent tasks from \mathbf{s} .

We recall that

$$J_T^A(\mathbf{s}, L) = \mathcal{C}^A(\mathbf{s}, L) + G_T^A(\mathbf{s}, L) \quad (157)$$

denotes the minimum average cost attained over T time slots by offloading L most imminent tasks from \mathbf{s} given the AMA's availability. From the definition of adjacent states in Definition 4, we have that $\bar{\mathbf{s}}_{a1L_a^*} \equiv \bar{\mathbf{s}}_{1(L_a^*-1)}$. As a result,

$$J_T^A(\mathbf{s}_a) - J_T^A(\mathbf{s}) = C_o. \quad (158)$$

Combining the above equality with Ineq. (154), we have

$$J_T^{\bar{A}}(\mathbf{s}_a) - J_T^{\bar{A}}(\mathbf{s}) < C_o. \quad (159)$$

Using the fact that \mathbf{s} is obtained by offloading the most imminent task from \mathbf{s}_a , the above inequality can be re-written as

$$J_T^{\bar{A}}(\bar{\mathbf{s}}_{a10}) < J_T^{\bar{A}}(\bar{\mathbf{s}}_{a11}) + C_o, \quad (160)$$

in which state $\bar{\mathbf{s}}_{a10}$ and $\bar{\mathbf{s}}_{a11}$ are obtained by offloading 0 task, and offloading the most imminent task from \mathbf{s}_a , respectively. From the definition of function $F^{\bar{A}}$ in Eq. (28), the above inequality is equivalent to

$$F^{\bar{A}}(T, \mathbf{s}_a, 1, 0) < F^{\bar{A}}(T, \mathbf{s}_a, 1, 1). \quad (161)$$

Now, combining Ineq. (161) with the convexity of function $F^{\bar{A}}$ presented in Lemma 3, we have

$$F^{\bar{A}}(T, \mathbf{s}_a, 1, 0) < F^{\bar{A}}(T, \mathbf{s}_a, 1, L) + LC_o, \text{ for all } L \in \mathbb{L}_1(\mathbf{s}_a). \quad (162)$$

This suggests that 0 is the optimal offloading decision associated with state \mathbf{s}_a for the given time horizon T , which is contradict to our assumption that \mathbf{s}_a is an offloading state. Therefore, we can make a conclusion that: As Ineq. (153) holds, \mathbf{s}_a is a non-offloading state.

To this end, the following has been proved: For two states \mathbf{s} and $\mathbf{s}_a \in \mathbb{S}_{adj}(\mathbf{s})$, state \mathbf{s}_a is a non-offloading state if and only if Ineq. (153) holds. As a consequence, \mathbf{s}_a is an offloading state if and only if $J_T(\mathbf{s}_a) - J_T(\mathbf{s}) \geq C_o$.

REFERENCES

- [1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.
- [3] G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Artificial intelligence enabled distributed edge computing for internet of things applications," in *2020 16th international conference on distributed computing in sensor systems (DCOSS)*. IEEE, May 2020, pp. 450–457.
- [4] P. Teymoori, T. D. Todd, D. Zhao, and G. Karakostas, "Efficient mobile computation offloading with hard task deadlines and concurrent local execution," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.
- [5] Y. Geng, Y. Yang, and G. Cao, "Energy-efficient computation offloading for multicore-based mobile devices," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 46–54.
- [6] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 81–93, 2014.
- [7] H. Huang, Q. Ye, and Y. Zhou, "Deadline-aware task offloading with partially-observable deep reinforcement learning for multi-access edge computing," *IEEE Transactions on Network Science and Engineering*, 2021.
- [8] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2017.
- [9] D. Van Le and C.-K. Tham, "An optimization-based approach to offloading in ad-hoc mobile clouds," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

- [10] —, “A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 760–765.
- [11] R. P. Stanley, *Catalan numbers*. Cambridge University Press, 2015.