GoPro: A Low Complexity Task Allocation Algorithm for a Mobile Edge Computing System

Arghyadip Roy
Mehta Family School of Data Science & Artificial Intelligence
Indian Institute of Technology Guwahati
Guwahati, India
arghyadip@iitg.ac.in

Nilanjan Biswas
Electronics and Communication Engineering Department
Birla Institute of Technology, Mesra
Jharkhand, India
nilanjan.biswas@bitmesra.ac.in

***Abstract*—In an Internet of Things (IoT) based network, tasks arriving at individual nodes can be processed in-device or at a local Mobile Edge Computing (MEC) server. In this paper, we focus on the optimal resource allocation problem for tasks arriving in an MEC based IoT network. To address the inherent trade-off between the computation time and the power consumption, we aim to minimize the average power consumption subject to a constraint on the deadline violation probability. The problem is formulated as a Constrained Markov Decision Process (CMDP) problem. To address the high complexities of achieving optimality, we propose a low-complexity heuristic task scheduling scheme. Efficacy of our approach is demonstrated using simulations.**

***Index Terms*—Mobile Edge Computing, IoT, CMDP.**

I. Introduction

Today's wireless communication systems are dominated by an enormous number of wireless devices with data-hungry applications. Moreover, these applications often require lots of processing power and resources as they are computationally intensive. Compute-intensive applications such as augmented and virtual reality, facial recognition, Internet of Things (IoT), Machine-to-Machine (M2M) communication are expected to be in great demand for users [1]. Next-generation communication networks, i.e., Fifth Generation (5G) and beyond 5G networks, are going to be the key enablers of many new services, e.g., industry 4.0, smart city, smart agriculture, which may be implemented with millions of IoT nodes [2]. As a part of Release 13 specifications, Third Generation Partnership Project (3GPP) [3] has standardized the operation of IoT devices in cellular bands and designated it as Narrow-Band (NB) IoT network operation [1], [4].

Event-driven tasks associated with completion deadlines may arrive at IoT devices (we call them IoT nodes) randomly. Depending on the application type (emergency/normal), task deadline may be hard or soft. In this paper, we focus on IoT tasks with soft deadlines [5, Table 2]. Typical IoT nodes have limited computation power, resulting in high deadline violation chances for in-device (also known by local) computation. To alleviate this problem, there is a provisioning for task offloading to a remote cloud server. Mobile edge computing (MEC) [6] server is another alternative for task computation. MEC servers are generally installed at local Access Points (APs), typically having less processing capability compared to the cloud server. However, MEC servers are often chosen over the cloud server due to their close proximity to IoT nodes, thereby reducing the round-trip delay significantly [7].

MEC servers can be potentially helpful for fast computation. However, the total power consumption (for offloading to MEC servers and computation thereafter) may be higher than in-device computation depending on the channel condition between IoT nodes and the MEC server. As tasks arrive randomly over time at IoT nodes and channel conditions of the IoT nodes with respect to the MEC server are also random, we target to capture the trade-off between power consumption and deadline violation from a overall system point of view over an *infinite horizon*. Since we consider tasks with soft deadline, in this paper, we aim to minimize the long-term average power consumption subject to a constraint on the average deadline violation probability (i.e., a constraint on the fraction of tasks violating the deadline). Since the optimal policy computation may be computationally demanding, we also propose a low-complexity task scheduling heuristic, motivated by practical considerations.

Related Work: Different standardizing bodies viz., European Telecommunications Standards Institute (ETSI) [6] and 3GPP (Release 17) [8] have indicated their interests towards incorporating the MEC technology as an integral part of 5G communication systems. Cloud functionalities in a limited form are implemented at MEC servers which are installed at the edge APs. Although the power consumption is low, local task computation may induce a large delay due to the limited computational ability of IoT nodes. To this end, it may be desirable from the overall system perspective that the IoT nodes offload their tasks to MEC servers to complete the tasks in due time. However, typical IoT nodes' power constraints put a limitation on their offloading capabilities. Power/ energy consumption and delay are considered as vital performance metrics in an MEC system. Related literature includes [9]–[15]. While [9]–[12] have considered a single wireless device, authors of [13]–[15] have considered multiple wireless devices for offloading to MEC servers. The computational resources at the MEC have been divided into multiple offloading wireless devices in [13]–[15]. The authors of [15] have considered full offloading of computational tasks and minimized the total energy consumption at wireless devices

2022 National Conference on Communications (NCC) | 978-1-6654-5136-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/NCC55593.2022.9806731

under various constraints. Both local and MEC computations have been considered in [9]–[11], [13], [14]. Both in [9], [10], the authors have constructed weighted utility function for optimizing energy consumption and delay under CPU power constraints.

Contributions: In this work, we consider an IoT network and focus on the optimal task allocation problem in a dynamic system where tasks arrive and depart dynamically. Specifically, we minimize the average power consumption for the IoT network while constraining the task deadline violation probability. We model the resource allocation problem as an infinite horizon average cost Constrained Markov Decision Process (CMDP) [16] problem. Moreover, to address the high complexity associated with the computation of optimal policy, we propose a low-complexity task allocation algorithm and demonstrate that it provides a near-optimal performance.

Although [9], [12] have considered the trade-off between the delay and the energy consumption, they focus on tasks arriving at a single wireless device. Contrary to [9], we consider dynamic arrival and departure of tasks at multiple IoT nodes. Moreover, instead of the delay parameter in [9], we take into account the deadline violation probability as a constraint since the former optimization problem may lead to unnecessary offloading to MEC server, leading to higher power consumption. Although [11], [15] address the power-delay trade-off by formulating the task scheduling problem as an MDP problem, instead of selective offloading considered in our paper, they have considered MEC offloading for each task which may result in a power-intensive solution. Moreover, the drift-plus-penalty method motivated by Lyapunov optimization adopted in [9], [11], [13] suffers from the $[O(1/V), O(V)]$ performance-delay trade-off (i.e., performance is within $O(1/V)$ of the optimal performance at the cost of delay of $O(V)$ for a given parameter $V \geq 0$) which is not the optimal trade-off [17]. Consideration of infinite CPU power (and consequently, negligible computation duration) at the MEC server which has been considered in [13], [14], may not be realistic. In this paper, we consider a finite CPU power which is of practical significance.

In a nutshell, the consideration of random task arrival, deadline violation constraint for IoT networks and limited CPU power at the MEC server make our system model realistic. The proposed CMDP problem can be solved to obtain the optimal solution. However, associated complexity may be high. The heuristic algorithm with low complexity proposed by us balances between the power consumption and the deadline violation probability by selective offloading of tasks in a probabilistic manner. To the best of our knowledge, this is the first work which considers the power-deadline violation trade-off in a dynamic system without any impractical assumption on CPU power availability at MEC server. Furthermore, we provide a low-complexity selective offloading scheme based on nodes' channel conditions.

The rest of the paper is as follows: Section II is on system model. We formulate the problem and provide the solution methodology in Section III. The low-complexity task allocation heuristic is proposed in Section IV. Sections V and VI describe experimental results and conclusions, respectively.

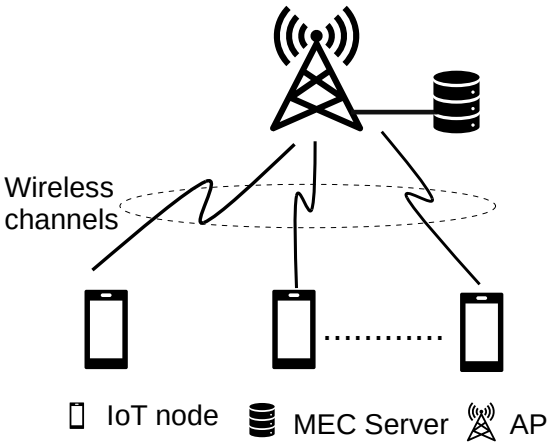


Figure 1: Logical system diagram.

II. System Model

We consider a system where an MEC server is present, as shown in Fig 1. The MEC server is connected to an AP using a lossless high-capacity link. IoT nodes are assumed to be stationary and can be present at any geographical point in an area. We assume that task arrivals at IoT nodes follow Poisson process with parameter λ. The service time of the tasks is assumed to be exponentially distributed with parameter μ. These assumptions are in accordance with [18]. The IoT network is assumed to operate in the licensed frequency band as provisioned by 3GPP standards such as NB-IoT [1], [4]. This enables the MEC server to distribute the computational resources uniformly to the IoT nodes in a centralized fashion.

Remark 1. *For the sake of the notational simplicity, we have considered a single MEC server system. However, the model can be easily extended to multiple MEC servers without any changes to the solution methodology. Since every IoT node can be mapped to the nearest MEC server based on the least power consumption, the multiple server case is reduced to a single server problem for every arriving task.*

A. State & Action Space

The system is modelled as a continuous time controlled stochastic process $\{X(t)\}_{t\geq 0}$ and is defined on a state space $\mathcal{S}$, say. Any state $s \in \mathcal{S}$ can be described as $s = (i, j_G, j_B)$, where i represents the number of active IoT in-device tasks, j_B and j_G represent the number of IoT tasks with bad and good channel conditions (with respect to the MEC server) performed in the MEC server, respectively. The system state does not change unless there is an arrival of a new task or a completion of an existing task, referred to as events. The events are arrival of a new task in an IoT node with good channel, arrival of a new task in an IoT node with bad channel, departure of a task from an IoT node, departure of a task from a good channel IoT node from MEC server and departure of a task from a

bad channel IoT node from MEC server, and we denote them by E_1, E_2, E_3, E_4 and E_5, respectively. Based on an event, an action is chosen and the system moves to a different state. Note that transition of $\{X(t)\}$ happens only when an event occurs, making it sufficient to observe the system state only at event epochs. We assume that $i \leq N_1$ and $(j_B + j_G) \leq N_2$, where N_1 and N_2 are large positive integers incorporated for analytical tractability. Note that in the state space, we need not consider channel conditions of nodes for local computation as the computation power and time do not depend on the channel condition, see (1) and (3).

The action space (say, $\mathcal{A}$) consists of a set of possible task scheduling strategies when a task arrives or departs. We denote the possible actions for arrivals, viz., in-device processing and task offloading to MEC server by A_1 and A_2, respectively. The feasible action in case of departure of a task is always A_0 (stay idle). Actions are chosen based on the system state and the event. We consider that actions A_1 and A_2 are feasible only when $i < N_1$ and $(j_B + j_G) < N_2$, respectively. When both conditions are violated, then the arriving task is dropped.

Remark 2. *3GPP standardizes 16 Channel Quality Indicators (CQIs), which can be divided into two groups, viz., good and bad. CQI of nodes are observed before choosing an action.*

Remark 3. *In case of departures from local nodes (MEC server), re-offloading of already offloaded MEC task (offloading to MEC server) can be considered. This may lead to an improvement in the performance. However, we do not incorporate this action in the action space to avoid the potential increase in control signing and session discontinuity.*

B. State Transitions

Based on the event (e) and the action ($a \in \mathcal{A}$), the system moves from state s to a different state (s') in a deterministic way. We describe these transitions in Table I. Based on the state-action pair, finite amounts of costs are incurred.

Table I: Transition Probability Table.

(e, a)	s'
(E_1, A_1)	$(i+1, j_G, j_B)$
(E_1, A_2)	(i, j_G+1, j_B)
(E_2, A_1)	$(i+1, j_G, j_B)$
(E_2, A_2)	(i, j_G, j_B+1)
(E_3, A_0)	$(i-1, j_G, j_B)$
(E_4, A_0)	(i, j_G-1, j_B)
(E_5, A_0)	(i, j_G, j_B-1)

C. Power Consumption Cost

Let the cost rate in terms of power consumption corresponding to state s, action a and event e be denoted by $c_p(s, a, e)$. For an IoT node, which has B bits for computation, power consumption may take place for local computation (A_1) or offloading (A_2).

1) Power consumption due to computation: The consumed power for computation is given by:

$$P_{c,y} = w_y f_y^3 \tag{1}$$

where $y \in \{\text{I,M}\}$ refer to in-device and MEC computation, respectively, w_y is the coefficient depending on chip architecture and f_y is the CPU frequency.

2) IoT node's Power consumption for offloading: The power consumption due to offloading to MEC server depends on the channel condition of the IoT node. Let the power consumption due to offloading of an IoT task with good (bad) channel condition to the MEC server be $P_{o,g}$ ($P_{o,b}$). Moreover, we assume that

$$P_{o,b} = P_{o,g} d, \tag{2}$$

where d is the channel degradation factor introduced to take into account the difference in channel gain for good and bad nodes (with channel coefficients h_g, h_b, say, respectively). In other words, $\frac{\mathbb{E}[|h_g|^2]}{\mathbb{E}[|h_b|^2]} = d$. Based on these, the cost rates due to power consumption are described in Table II.

Table II: Power Consumption Cost Function Table.

(e, a)	$c_p(s, a, e)$
(E_1, A_1)	$P_{c,I}(i+1) + P_{c,M}(j_B + j_G) + P_{o,g}(j_G + dj_B)$
(E_1, A_2)	$P_{c,I}(i) + P_{c,M}(j_B + j_G + 1) + P_{o,g}(j_G + 1 + dj_B)$
(E_2, A_1)	$P_{c,I}(i+1) + P_{c,M}(j_B + j_G) + P_{o,g}(j_G + dj_B)$
(E_2, A_2)	$P_{c,I}(i) + P_{c,M}(j_B + j_G + 1) + P_{o,g}(j_G + d(j_B + 1))$
(E_3, A_0)	$P_{c,I}(i-1) + P_{c,M}(j_B + j_G) + P_{o,g}(j_G + dj_B)$
(E_4, A_0)	$P_{c,I}(i) + P_{c,M}(j_B + j_G - 1) + P_{o,g}(j_G - 1 + dj_B)$
(E_5, A_0)	$P_{c,I}(i) + P_{c,M}(j_B + j_G - 1) + P_{o,g}(j_G + d(j_B - 1))$

D. Deadline Violation Cost

Let the cost rate in terms of deadline violation corresponding to state s, action a and event e be denoted by $c_d(s, a, e)$. Let the local computation time and MEC computation time be denoted by $T_{c,I}$ and $T_{c,M}(s)$, respectively. Hence,

$$T_{c,I} = \frac{BC}{f_I}, \tag{3}$$

$$T_{c,M}(s) = \frac{B}{R_m} + \frac{BC(j_G + j_B)}{f_M} \tag{4}$$

where $R_m = W \log_2(1 + \frac{P_{o,b}|h_b|^2}{N_0}) = W \log_2(1 + \frac{P_{o,g}|h_g|^2}{N_0})$ (Using (2)), W is the channel bandwidth, N_0 is the noise power and C is required CPU cycles to compute a bit. The term $(j_G + j_B)$ in (4) appears due to equal allocation of CPU resources among MEC tasks. The first term in (4) signifies the task offloading time.

The instantaneous deadline violation cost rate is

$$c_d(s, a, e) = \begin{cases} 1, \ T_{c,I} > L, a = A_1, \\ 1, \ T_{c,M}(i, j_G + 1, j_B) > L, (a, e) = (A_2, E_1), \\ 1, \ T_{c,M}(i, j_G, j_B + 1) > L, (a, e) = (A_2, E_2), \\ 0, \ \text{otherwise} \end{cases}$$

where L is the deadline associated with a task. The deadline violation cost is unity when the deadline of arriving task under the chosen action is not met, zero otherwise.

III. Problem Setup

A task offloading *policy* is a sequence of rules regarding the actions to be chosen at different states and decision epochs. We aim to determine the optimal task scheduling policy which minimizes the power consumption subject to a constraint on the deadline violation probability. This problem can be formulated as a CMDP problem for which there exists an optimal *stationary randomized* policy [16]. A stationary randomized policy is a policy where the action probabilities at a given state do not change with time. A special case of a stationary randomized policy is a stationary deterministic policy which chooses a particular action deterministically in a given state. The problem is a continuous time CMDP problem as arrivals and departures can happen anytime.

A. Problem Formulation

Since the zero state can be reached from any state with a non-zero probability, the Markov chains associated with stationary policies are unichain to guarantee a unique stationary distributions. Let $\mathcal{P}$ be the set of all stationary policies, and the average power consumption cost and deadline violation cost under the policy $P \in \mathcal{P}$ be denoted by C^P and D^P, respectively. The CMDP problem can be described as follows:

$$\begin{aligned} &\text{Minimize: } C^P = \lim_{t\to\infty} \frac{1}{t}\mathbb{E}_P[C_1(t)], \\ &\text{subject to: } D^P = \lim_{t\to\infty} \frac{1}{t}\mathbb{E}_P[C_2(t)] \leq C_{\max}, \end{aligned} \tag{5}$$

where $C_1(t)$ and $C_2(t)$ denote the total costs due to power consumption and deadline violation till time t, respectively, and $C_{\max}$ denotes the upper bound on the average deadline violation probability. Note that the expectation is taken over the policy. We aim to determine the optimal policy for the CMDP problem.

B. Solution Methodology

Optimal policy can be computed using Value Iteration Algorithm (VIA) on the value function of states [19] and gradient descent [16] to handle the constraint. Dynamic Programming (DP) techniques such as VIA can be applied to solve this continuous time MDP problem. Before that, it has to be converted to an equivalent discrete time MDP problem using a technique called uniformization [19] to ensure both discrete time and continuous time models have the same expected cost for a given stationary policy.

Let $\tau(s,a,e)$ denote the expected time until the next event if action a is chosen in state s in response to event e. We choose $0 < \delta \ll min_{s,a,e}\tau(s,a,e)$. Keeping the state and action space unchanged, let $\hat{p}(s,e)$, $\hat{c}_p(s,a,e)$ and $\hat{c}_d(s,a,e)$ denote the event probabilities in state s, cost due to power consumption and cost due to deadline violation, respectively, in the transformed model. Applying uniformization technique, we obtain, $\hat{c}_p(s,a,e) = c_p(s,a,e)$ and $\hat{c}_d(s,a,e) = c_d(s,a,e)$. Furthermore, $\hat{p}(s,E_1) = \lambda p_g \delta$, $\hat{p}(s,E_2) = \lambda(1-p_g)\delta$, $\hat{p}(s,E_3) = i\mu\delta$, $\hat{p}(s,E_4) = j_G\mu\delta$, $\hat{p}(s,E_5) = j_B\mu\delta$, where p_g denote the probability that the arriving task in the IoT node observe a good channel with respect to the MEC server.

After we have obtained an equivalent discrete time MDP model, we fix the Lagrange Multiplier (LM) β to write an equivalent unconstrained cost function

$$\hat{c}(s,a,e;\beta) = \hat{c}_p(s,a,e) + \beta\hat{c}_d(s,a,e).$$

Then, the DP equation $\forall s, s' \in \mathcal{S}$ is given by

$$\begin{aligned} V(s) = &\sum_e \hat{p}(s,e) \min_a [\hat{c}(s,a,e;\beta) + V(s'(s,e,a))] \\ &+ \big(1 - \sum_e \hat{p}(s,e)\big)V(s), \end{aligned}$$

where $V(s)$ is the value function of state s and $s'(s,e,a)$ is the next state when action a is chosen in state s for event e. The optimal LM β^* can be computed iteratively using gradient descent algorithm [20], as follows:

$$\beta_{k+1} = \beta_k + \frac{1}{k}(C^\pi_{\beta_k} - C_{\max}), \tag{6}$$

where $C^\pi_{\beta_k}$ is the average deadline violation cost associated with the policy π under LM value β_k at k^{th} iteration. For a fixed β, the iterative VIA scheme is as follows:

$$\begin{aligned} V_{n+1}(s) = &\sum_e \hat{p}(s,e) \min_a [\hat{c}(s,a,e;\beta) + V_n(s'(s,e,a))] \\ &+ \big(1 - \sum_e \hat{p}(s,e)\big)V_n(s), \end{aligned}$$

where $V_n(.)$ is the value function estimate after n iterations. After determining β^*, the optimal policy for the CMDP can be computed as a mixture of two pure policies $\pi_{\beta^*-\epsilon}$ and $\pi_{\beta^*+\epsilon}$ obtained by a perturbation of ϵ in both directions of β^*. Let the average cost associated with these policies be $C^\pi_{\beta^*-\epsilon}$ and $C^\pi_{\beta^*+\epsilon}$. We determine the value of p such that

$$pC^\pi_{\beta^*-\epsilon} + (1-p)C^\pi_{\beta^*+\epsilon} = C_{\max}.$$

The randomized optimal policy for the considered CMDP problem is a policy where actions corresponding to pure policies $\pi_{\beta^*-\epsilon}$ and $\pi_{\beta^*+\epsilon}$ are chosen with probabilities p and $(1-p)$, respectively.

IV. Proposed Task Scheduling Algorithm

The CMDP formulation described in the last section can be solved by DP methods, however often at the cost of high computation complexity. For example, policy iteration [19] which is a well-known DP method, is associated with a computational complexity of $O(|\mathcal{A}|^{|\mathcal{S}|})$. This is known as the *curse of dimensionality*. To address this issue, we propose a low-complexity heuristic motivated by the practical considerations. Contrary to DP methods, this method does not require any offline computation and hence, is practically implementable.

A. Good Offload Bad Probabilistic Offload Algorithm

In this section, we propose a heuristic which offloads all tasks with good channel (with respect to MEC server) to the MEC server. However, if there is an arrival of a task with bad channel (with respect to MEC), then the task is probabilistically offloaded to the MEC server. The motivation behind

such a policy is that the offload of good channel tasks to the MEC server should have more priority than that of bad channel tasks as the bad channel tasks are prone to consume more power for offloading. Therefore, good and bad channel tasks are offloaded to the MEC server deterministically and with a finite probability, respectively. Since excessive offloading to the MEC server may lead to high power consumption, the probability of bad task offload is increased as the number of IoT in-device tasks increases. This strategy helps in balancing between the power consumption and the deadline violation probability. The details of the proposed **G**ood **O**ffload-Else-**Pr**obabilistic-**O**ffload (**GoPro**) are given in Algorithm 1.

Algorithm 1 Good Offload Else Probabilistic Offload (GoPro) Algorithm.

1: **while** TRUE **do**
2: Determine event e in the current decision epoch.
3: **if** $(e = E_1)$ **then**
4: Select action $a = A_2$ (offloading to MEC server).
5: **else if** $(e = E_2)$ **then**
6: Determine the current system state.
7: **if** Number of in-device tasks$= i$ **then**
8: **procedure** PROB–OFFLOAD
9: Take action A_2 with probability $f(i)$ and action A_1 with probability $(1 - f(i))$.
10: **end procedure**
11: **end if**
12: **else**
13: Take action A_0.
14: **end if**
15: **end while**

We first determine the event e in the current decision epoch. If there is an arrival of a good channel task (event E_1), then the task is offloaded to the MEC server (action A_2). This strategy is adopted with a view that the good channel tasks consume less power than the bad channel tasks, when offloaded. Hence, good channel tasks are always offloaded to the MEC server whenever capacity is available in the MEC server (Line 4). On the other hand, if the current event is an arrival of a bad channel task, then the task is probabilistically offloaded to the MEC server. If the current system state is (i, j_G, j_B), then the task is offloaded to the MEC server (action A_2) with probability $f(i)$, where $0 \leq f(i) \leq 1$, and $f(i)$ is a non-decreasing function in i (Line 8). With the remaining probability $(1-f(i))$, the task is computed locally (referred to as PROB_OFFLOAD). The justification behind this strategy is as follows. Since tasks with bad channel consume more power than good channel tasks, we initially process them locally with high probability so that the power consumption can be made low. However, excessive local computation may lead to an increase in the deadline violation probability. Hence, as i increases, $f(i)$ is gradually increased. As i increases, the probability of MEC offloading (choosing action A_2) for event E_2 is increased to maintain a balance between the power consumption and the deadline violation probability. This is in line with our system objective. When departure occurs, the only feasible action is action A_0 (Line 13).

B. Complexity Analysis

In this section, we analyze the storage and computational complexities associated with the proposed algorithm and compare with those of the optimal policy. In the case of the optimal policy, we need a storage of $O(|\mathcal{S}|)$ as we need to store the optimal action corresponding to each state. The worst case computational complexity associated with policy iteration [19] is $O(|\mathcal{A}|^{|\mathcal{S}|})$ which is same as the total number of feasible policies. In case of GoPro, the computational complexity is $O(1)$ as we just need to toss a coin and decide whether action A_1 or A_2 needs to be chosen based on the value of $f(i)$ (in case of event E_2). In case of other events, no computation is involved. Since we need the knowledge of the current value of i (number of active IoT in-device tasks) and corresponding values of $f(i)$, the storage complexity is $O(N_1)$ as $i \leq N_1$ (see Section IIA).

Therefore, both the storage and computational complexities of GoPro are much better than the optimal policy. Moreover, the proposed algorithm can be implemented online as it does not require the prior knowledge of system parameters.

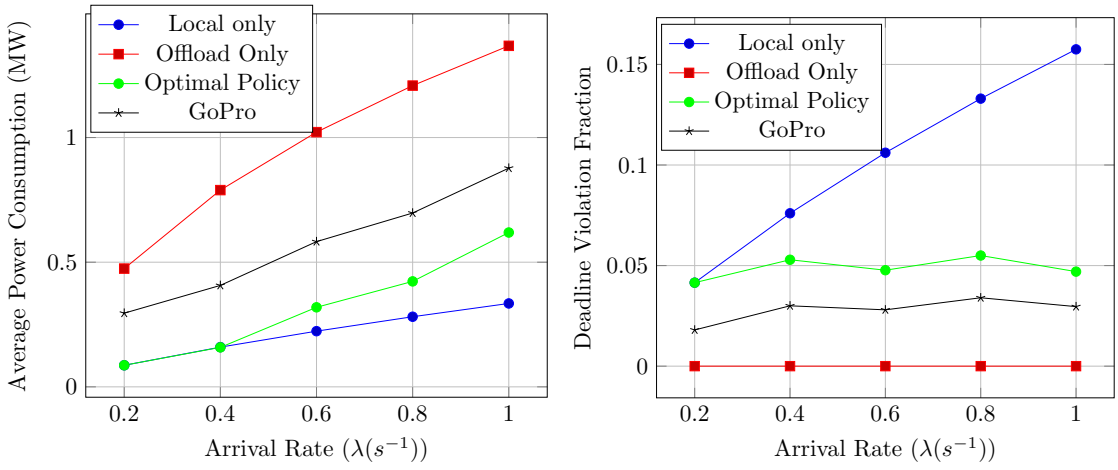
V. Numerical Results and Discussions

In this section, we demonstrate the performances of different schemes and compare with our proposed algorithm in terms of the average power consumption and the deadline violation probability. The experiments are conducted in MATLAB. The average costs of different policies in terms of the average power consumption and the deadline violation probability are computed. We compare the performances of GoPro with the optimal policy, local only (all tasks are computed locally) and offload only (all tasks computed at the MEC server) policies. The values of different parameters considered by us are shown in Table III, following [1]. We take $N_1 = N_2 = 10$, $E[|h_g|^2] = 0.19$ (exponential distribution), $P_{o,g} = 0.5$, $d = 2$, $L = 0.001$ s and $C_{\max} = 0.05$. We also choose $f(i) = 0$ for $i < 4$ and $f(i) = 1$, otherwise.

Table III: System Parameters.

Parameters	Values	Parameters	Values
w_I	10^{-30}	w_M	10^{-30}
f_I	10^7 Hz	f_M	10^{12} Hz
B	512 bits	C	10^4 CPU/bits
N_0	-132.24 dBm	W	1.4 MHz

As observed in Fig. 2a and 2b, when all the tasks are offloaded to the MEC server, the delay and consequently, the deadline violation fraction are low at the cost of high average power consumption. On the contrary, the local task computation results in low power consumption at the cost of high deadline violation fraction. The optimal policy strikes a balance between these two parameters by solving a constrained MDP problem. The low complexity heuristic GoPro perform similar to the optimal policy. Compared to optimal policy, GoPro is more conservative in terms of deadline violation



(a) (b)

Figure 2: Plot of different parameters vs. arrival rate:(a) average power consumption, (b) deadline violation fraction ($\mu = 1$).

probability at the cost of higher power consumption. This happens because the GoPro scheme is designed with a view of creating a balance between the power consumption and the deadline violation probability.

VI. Conclusions

In this paper, we have formulated the task allocation problem in an MEC based IoT network as a CMDP problem which minimizes the average power consumption subject to a constraint on the deadline violation probability. Since the computation of optimal policy is of exponential complexity, we have proposed a low complexity heuristic GoPro which balances between power consumption and deadline violation. GoPro demonstrates similar trend as that of the optimal policy, outperforming existing algorithms. The tuning of offloading probability parameters as a function of task arrival rates remains an important future direction.

Acknowledgement

The work of A. Roy is supported by the Start-up Grant at IIT Guwahati.

References

[1] A. Hoglund, X. Lin, O. Liberg, A. Behravan, E. A. Yavuz, M. Van Der Zee, Y. Sui, T. Tirronen, A. Ratilainen, and D. Eriksson, "Overview of 3GPP release 14 enhanced NB-IoT," *IEEE network*, vol. 31, no. 6, pp. 16–22, 2017.

[2] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5G era: Enablers, architecture, and business models," *IEEE journal on selected areas in communications*, vol. 34, no. 3, pp. 510–527, 2016.

[3] 3GPP RP-161901, "Revised Work Item Proposal: Enhancements of NB-IoT," 2016.

[4] L. Feltrin, G. Tsoukaneri, M. Condoluci, C. Buratti, T. Mahmoodi, M. Dohler, and R. Verdone, "Narrowband IoT: A survey on downlink and uplink perspectives," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 78–86, 2019.

[5] S. Malik, S. Ahmad, I. Ullah, D. H. Park, and D. Kim, "An adaptive emergency first intelligent scheduling algorithm for efficient task management and scheduling in hybrid of hard real-time and soft real-time embedded IoT systems," *Sustainability*, vol. 11, no. 8, p. 2192, 2019.

[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[7] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[8] S. Chitturi. Enabling edge computing applications in 3gpp. [Online]. Available: https://www.3gpp.org/news-events/2152-edge-sa6

[9] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, 2018.

[10] N. Biswas, H. Mirghasemi, and L. Vandendorpe, "Joint optimization of relaying rate and energy consumption for cooperative mobile edge computing," in *IEEE WiOPT*, 2020, pp. 1–8.

[11] D. Han, W. Chen, and Y. Fang, "Joint channel and queue aware scheduling for latency sensitive mobile edge computing with power constraints," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3938–3951, 2020.

[12] Z. Li, V. Chang, J. Ge, L. Pan, H. Hu, and B. Huang, "Energy-aware task offloading with deadline constraint in mobile edge computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, pp. 1–24, 2021.

[13] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *IEEE GLOBECOM*, 2016, pp. 1–6.

[14] 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.

[15] M. Merluzzi, P. Di Lorenzo, S. Barbarossa, and V. Frascolla, "Dynamic computation offloading in multi-access edge computing via ultra-reliable and low-latency communications," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 342–356, 2020.

[16] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[17] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[18] T. Bonald and J. W. Roberts, "Internet and the Erlang formula," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 1, pp. 23–30, 2012.

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

[20] A. Roy, P. Chaporkar, and A. Karandikar, "Optimal radio access technology selection algorithm for LTE-WiFi network," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6446–6460, 2018.