Contextual Multi-Armed Bandits for Non-Stationary Heterogeneous Mobile Edge Computing

Maximilian Wirth, Andrea Ortiz, and Anja Klein
Communications Engineering Lab, Technische Universität Darmstadt, Germany
{m.wirth, a.ortiz, a.klein}@nt.tu-darmstadt.de

***Abstract*—Base station (BS) selection for task offloading in Mobile Edge Computing (MEC) is a challenging problem due to the dynamic nature of MEC systems. The wireless channel as well as the load of BSs are stochastic quantities that can change in a statistically non-stationary fashion. Moreover, the computation capabilities of the BSs are heterogeneous. As the dynamic behaviour of a MEC system is, in practical scenarios, not known in advance, deciding where to offload has to be done under uncertainty about the MEC system and considering its non-stationary and heterogeneous characteristics. This paper investigates latency minimization in MEC with heterogeneous BSs. In order to meet low latency demands, a mobile unit (MU) has to quickly identify the best BS for offloading different computation tasks while facing uncertainty about the non-stationary system dynamics. To solve this problem, we propose a novel piece-wise stationary contextual Multi-Armed Bandit (MAB) algorithm that treats different task types as context and detects non-stationary changes in the BSs' performance. With the use of extensive simulations, we show that our proposed approach outperforms state-of-the-art algorithms, as it quickly adapts to changes in the MEC system and exhibits no penalty during stationary phases.**

I. Introduction

Many mobile applications, such as video and image processing, mobile gaming or real-time face recognition, are computationally demanding and latency sensitive. However, the processing power of the existing battery powered mobile units (MUs) is usually limited. Mobile Edge Computing (MEC) [1], [2] promises to be an enabler for the timely execution of such applications as it brings large computation resources to the edge of the network. By offloading a computation task to a base station (BS) equipped with a co-located edge server, both, the computation time and the energy consumption of the MU, can be reduced significantly.

Deciding to which BS a computation task should be offloaded is a non-trivial problem because the MEC systems are dynamic. This dynamic nature of MEC is caused by the variability of the wireless channel used for the communication between MUs and BSs, as well as the changing loads and configurations of the BSs. Moreover, these changes can occur in a statistically non-stationary fashion. For example, a non-stationary change of the wireless channel might be caused by temporary blockages of transmission paths, or a non-stationary event in the BS's load might originate from an abrupt increase in the number of connected users. Furthermore, the computation capabilities of the BSs are heterogeneous in the sense that different BSs are equipped with different hardware [3] and software [4]. Thus, depending on the individual hardware and software requirements of the task to be computed, the computation performance of the BSs can vary greatly. The dynamic behaviour of a MEC system is, in practical scenarios, not known in advance. Therefore, deciding where to offload has to be done under uncertainty about the MEC system and considering its non-stationary and heterogeneous characteristics.

Many works have treated the subject of BS selection for task offloading in MEC from an optimization point of view [5], [6]. These works overcome the challenges in dynamic MEC systems by making the unrealistic assumption that perfect knowledge about the statistics of the dynamic MEC system is available. With the goal of minimizing the latency or energy consumption, they solve complex non-convex or even NP-hard optimization problems by applying relaxation techniques or heuristic methods. Although these approaches provide an upper bound of the performance, they cannot be deployed in real MEC systems, because finding the best offloading decision is usually computationally prohibitive and therefore, time-consuming, which contradicts the low latency requirement imposed by many mobile applications. Other recent works have tackled the offloading problem in dynamic MEC with a reinforcement learning approach [7]–[9]. Reinforcement learning is better suited for cases in which the statistically (non-)stationary MEC system dynamics are not known. However, these works make simplifying assumptions regarding the characteristics of the MEC system dynamics. The authors in [7] consider a fully stationary scenario where all the BSs have the same computation capabilities. In [8], only non-stationary server load distributions are considered while the dynamics of the wireless channels of the BSs are ignored and [9] fails to address heterogeneity, i.e., the task dependent computation capabilities of the BSs.

In this work, we investigate a statistically non-stationary MEC system with heterogeneous BSs. We consider a single MU and multiple BSs with co-located edge servers. For every new task to be computed, the MU selects one BS for task offloading with the goal of minimizing the experienced latency. As in practical systems, no prior knowledge about any system information, such as expected transmission rates, server loads or task execution latencies, is assumed. Our contributions are:

This work has been funded by the BMBF project Open6GHub, grant number 16KISK014, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projektnummer 210487104 - SFB 1053 MAKI, and the LOEWE Center EmergenCITY.

- We model a realistic MEC system in a highly dynamic non-stationary environment in which the wireless channel as well as the load of BSs and edge servers are modelled by piece-wise stationary random variables.
- We consider heterogeneous BSs, i.e., each BS has different hardware and software configurations. Thus, the task execution latency depends on the specific hardware and software requirements of the task to be computed and is different for each BS.
- We propose C-CPD-UCB, a novel reinforcement-learning algorithm based on contextual Multi-Armed Bandits (MABs) that detects change points in the BSs' performance for a swift adaptation to non-stationary environments while exhibiting negligible performance penalties during stationary phases. Our algorithm is able to handle different types of tasks by modeling them as its context. This allows us to identify the best BS depending on the hardware and software requirements of each task.
- We carry out extensive simulations to verify the superior performance of our proposed algorithm.

The remainder of the paper is structured as follows. In Section II, we introduce our system model while in Section III, the optimization problem for the latency minimization is formulated. In Section IV, our proposed algorithm is explained. The performance of our solution is evaluated in Section V and Section VI concludes the paper.

II. System Model

A. Overview

The considered scenario consists of a single MU and $K \in \mathbb{N}$ BSs in the set $\mathcal{K} = \{1, \dots, K\}$. Each BS is equipped with an adjacent edge server for task computation. In every time step $t = 1, \dots, T$, where $T \in \mathbb{N}$ is the number of considered time steps, the MU has a task to be computed. At the beginning of each time step t, the MU selects one BS $k \in \mathcal{K}$ for task offloading. Different tasks can have different software or hardware requirements, e.g., processor, memory, graphics card. Therefore, each task is characterized by a task type α_t, taken from the set $\mathcal{A} = \{1, \dots, A\}$ consisting of $A \in \mathbb{N}$ different task types. It is assumed that each task type has similar requirements in terms of hardware and software. Examples for task types are video-processing, gaming or augmented reality. Let $T_{k,t}^{\text{exe}}(\alpha_t)$ be the task execution latency associated with selecting BS k in time step t, i.e., the time needed to complete the task execution on the edge server of BS k. Due to the specific hardware and software requirements of task type α_t, $T_{k,t}^{\text{exe}}(\alpha_t)$ depends heavily on the selected BS's software and hardware set-up. As discussed in [10], it is generally hard to determine the exact execution latency of a task, even when the deployed hardware and software are known. Moreover, the MU may not have knowledge about each BS's exact hardware and software capabilities. Thus, it is assumed that the MU is uncertain about the exact task execution latency and only knows the type α_t of each task. In addition to the type, each task is described by a task size $\beta_t(\alpha_t)$ in bits, that the MU has to transmit to the BS. The task size $\beta_t(\alpha_t)$ is known to

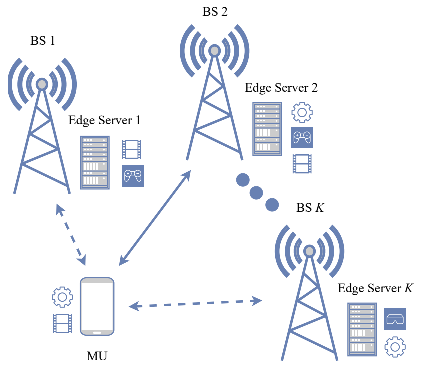

Fig. 1. MEC scenario with heterogeneous edge servers.

the MU and depends on the type α_t of the task. Note that the task size does not affect the task execution latency $T_{k,t}^{\text{exe}}(\alpha_t)$.

The length of a single time step is variable and equal to the task latency. The task latency when selecting BS k in time step t can be expressed by

$$T_{k,t}(\alpha_t) = T_{k,t}^{\text{tx}}(\alpha_t) + T_{k,t}^{\text{wait}} + T_{k,t}^{\text{exe}}(\alpha_t), \tag{1}$$

where $T_{k,t}^{\text{tx}}(\alpha_t)$ is the transmission latency, associated with the task transmission to the BS k in time step t and $T_{k,t}^{\text{wait}}$ denotes the waiting latency, attributed to the time the task has to wait in the queue of the edge server of BS k in time step t. As most other works, the time required to send the result back to the MU is assumed to be negligible [4].

It is assumed that the K BSs are positioned at fixed locations for all considered time steps. Similar to [11], only small and slow movements of the MU are considered, i.e., the distance between the MU and each BS stays approximately constant. However, the wireless channel between the MU and the BSs can change significantly due to movements from line-of-sight (LoS) to no-line-of-sight (NLoS) or due to changes in the surrounding environment, like temporary blockages.

B. Communication Model

The channel coefficient $h_{k,t}$ of the wireless channel from MU to BS k in time step t is drawn from a Rician distribution, which allows to capture both, LoS and NLoS transmission conditions. Consequently, $h_{k,t}$ can be expressed as [12]

$$h_{k,t} = \sqrt{\frac{\kappa_k}{\kappa_k + 1}} e^{j\theta_{k,t}} + \sqrt{\frac{1}{\kappa_k + 1}} \tilde{h}_{k,t}, \tag{2}$$

where $\tilde{h}_{k,t}$ follows a zero-mean and unit-variance complex Gaussian distribution and the phases $\theta_{k,t}$ are uniformly distributed on $[0, 2\pi]$ and statistically independent. Moreover, $\kappa_k \geq 0$ denotes the energy ratio between LoS and NLoS for the wireless channel between MU and BS k. It is assumed that κ_k can change over time in a piece-wise constant fashion, i.e., κ_k changes at certain time steps and stays constant in between. This variation is caused by MU movements and implies that $h_{k,t}$ is drawn from a piece-wise stationary probability distribution. The channel gain of the wireless link from MU to BS k in time step t is given by [12]

$$H_{k,t} = d_{k,t}^{-\gamma/2} \cdot h_{k,t}, \tag{3}$$

where $d_{k,t}$ is the distance between MU and BS k in time step t and $d_{k,t}^{-\gamma/2}$ is the path loss with path loss coefficient $\gamma \geq 2$.

It is considered that each BS uses a separate frequency band and an orthogonal frequency-division multiple-access scheme for the communication with the MU and other connected users. Hence, the signals of the MU and other users do not interfere with each other. In order to capture the impact of other users on the load of the wireless network, let $N_{k,t}^{\text{tx}}$ be the number of additional users connected to BS k in time step t. $N_{k,t}^{\text{tx}}$ is assumed to be drawn form a Poisson distribution $\text{Pois}(\nu_k)$ with parameter $\nu_k > 0$. Using Shannon's channel capacity formula, the maximum achievable transmission rate from MU to BS k in time step t can be expressed by

$$R_{k,t} = \frac{B}{1 + N_{k,t}^{\text{tx}}} \cdot \log_2\left(1 + \frac{|H_{k,t}|^2 \cdot P}{\sigma_n^2}\right) \quad \text{in } \frac{\text{bit}}{\text{s}}, \tag{4}$$

where P is the constant transmit power of the MU, σ_n^2 denotes the thermal noise power and $B > 0$ denotes the maximum system bandwidth, which is shared among the MU and other users connected to the same BS. Furthermore, it is assumed that ν_k is piece-wise constant, rendering the allocated bandwidth a piece-wise stationary random variable. This allows us to capture the dynamically varying load of the BSs. The transmission latency associated with the transmission of a task to BS k is given by

$$T_{k,t}^{\text{tx}}(\alpha_t) = \frac{\beta_t(\alpha_t)}{R_{k,t}}. \tag{5}$$

C. Computation Model

As mentioned before, the exact task execution latency $T_{k,t}^{\text{exe}}(\alpha_t)$ is not precisely known to the MU. In order to account for variability within the same task type and changing availability of the BSs' resources, $T_{k,t}^{\text{exe}}(\alpha_t)$ is drawn from an exponential distribution $\text{Exp}(\eta_k(\alpha_t))$ with distribution parameter $\eta_k(\alpha_t) > 0$ for BS k and task type α_t. Similar to [13], the waiting latency in the server queue is defined as

$$T_{k,t}^{\text{wait}} = \sum_{n=1}^{N_{k,t}^{\text{queue}}} T_{k,n,t}^{\text{queue}}(\phi_{k,n,t}), \tag{6}$$

where the number of tasks $N_{k,t}^{\text{queue}}$ in the queue of the edge server of BS k in time step t is drawn from a Poisson distribution $\text{Pois}(\lambda_k)$ with parameter $\lambda_k > 0$. Furthermore, $T_{k,n,t}^{\text{queue}}(\phi_{k,n,t})$ denotes the execution latency and $\phi_{k,n,t}$ denotes the task type of the n-th task in the queue of the edge server of BS k in time step t. The task types $\phi_{k,n,t}$ are selected randomly from $\mathcal{A}$ with equal probability and $T_{k,n,t}^{\text{queue}}(\phi_{k,n,t})$ is drawn from $\text{Exp}(\eta_k(\phi_{k,n,t}))$.

It is assumed that the parameters λ_k and $\eta_k(\alpha)$ can change over time, such that the waiting latency $T_{k,t}^{\text{wait}}$ and task execution latency $T_{k,t}^{\text{exe}}(\alpha_t)$ are drawn from piece-wise stationary distributions. These non-stationary changes can be caused by load changes on the BSs' edge servers as well as alterations in their software and hardware configurations.

D. Piece-Wise Stationary Model

The task latency $T_{k,t}(\alpha_t)$ associated with offloading a task to BS k in time step t is a sample drawn from a distribution that is characterized by the parameters ν_k, κ_k, λ_k and $\eta_k(\alpha_t)$. More specifically, the task latency is a realization of the random variable $T_k(\alpha)$, which describes the random task latency that the MU experiences when it offloads a task of type $\alpha \in \mathcal{A}$ to BS $k \in \mathcal{K}$. Furthermore, it is assumed that the task latency is upper bounded by $T_{\max} > 0$, i.e., $T_{k,t}(\alpha_t) \in (0, T_{\max}]$. Moreover, let $\mu_k(\alpha) = \mathbb{E}[T_k(\alpha)]$ be the expected task latency that is associated with selecting BS k for the task type α. Finally, due to the piece-wise constant nature of the distribution parameters ν_k, κ_k, λ_k and $\eta_k(\alpha)$, the random variable $T_k(\alpha)$ is piece-wise stationary, i.e., $\mu_k(\alpha)$ changes over time. It is assumed that the distributions of the task latency random variables $T_k(\alpha)$, for all BSs $k \in \mathcal{K}$ and task types $\alpha \in \mathcal{A}$ combined, do not change more than $\Gamma(T) \in \mathbb{N}$ times until the end of the last time step T, where $\Gamma(T) \ll T$.

III. Problem Formulation

The MU's goal is to select the BSs that minimize the sum of all experienced task latencies. The optimization problem can be written as

$$\min_{\{y_{t,k}\}_{k\in\mathcal{K},\ t\in\{1,\ldots,T\}}} \sum_{t=1}^{T}\sum_{k=1}^{K} y_{t,k} T_{k,t}(\alpha_t) \tag{7}$$

$$\text{subject to} \sum_{k=1}^{K} y_{t,k} = 1, \qquad \forall t, \tag{8}$$

$$y_{t,k} \in \{0,1\}, \qquad \forall k,\ \forall t, \tag{9}$$

where $y_{t,k}$ is a decision variable, i.e., $y_{t,k} = 1$ if MU selects BS k in time step t and $y_{t,k} = 0$ otherwise. Moreover, Constraint (8) ensures that only one BS is selected in every time step and Constraint (9) enforces the binary nature of the decision variable. It is worth noting that decisions made in different time steps have no impact on each other. Thus, the BS selection can be performed for each time step independently, i.e., in each time step t, the MU selects a BS such that the task latency $T_{k,t}(\alpha_t)$ for the given task type α_t is minimized. Furthermore, the solution of the aforementioned optimization problem requires non-causal knowledge about the dynamic behaviour of the MEC system.

IV. C-CPD-UCB Algorithm

A. Overview

As in real MEC systems the MU has no prior knowledge about the statistical behaviour of the wireless channel, the BSs' loads or their computation capabilities, in this section, we present *Contextual Change-Point-Detection Upper-Confidence-Bound* (C-CPD-UCB), a learning-based approach to learn the expected task latency $\mu_k(\alpha)$ associated with each BS. C-CPD-UCB is based on the Upper-Confidence-Bound algorithm (UCB) [14]. Since UCB is designed for stationary bandits, C-CPD-UCB is able to achieve near optimal performance during stationary phases. Moreover, it takes into account that the probability distributions for the task latencies $T_k(\alpha)$ are piece-wise stationary, i.e., $\mu_k(\alpha)$ changes over time, and allows the MU to identify and adapt to these changes.

C-CPD-UCB is a MAB approach. In the MAB-setting, the learning agent is the MU and the arms of the bandit, or actions, correspond to the K BSs in $\mathcal{K}$. Furthermore, with a slight abuse of the terminology, the rewards are defined as the experienced task latencies $T_{k,t}(\alpha_t)$ and our goal is to minimize the reward in each time step t. As the experienced task latency $T_{k,t}(\alpha_t)$ depends on the task's hardware and software requirements, i.e., the task type α_t, the set of task types $\mathcal{A}$ is considered as the context space in our MAB formulation. It is worth noting that in our case, the context space is one-dimensional and discrete. Nevertheless, our proposed algorithm can also be extended to multi-dimensional and continuous context spaces.[1] Incorporating context into existing algorithms for non-stationary bandits is not straightforward. For contextual MABs, every arm and context combination needs a separate estimate for the expected reward. At the same time, compared to non-contextual MABs, there are less reward samples for every estimate available given the same number of time steps. This results in an increase of the convergence time and a harder detection and adaptation to changes in the reward distributions.

B. MAB Formulation

Our proposed algorithm is summarized in Algorithm 1. In accordance to the UCB [14] algorithm, at each time step t, the MU selects a BS $\hat{k}_t$ according to

$$\hat{k}_t = \arg\min_{k\in\mathcal{K}} \hat{\mu}_{k,t-1}(\alpha_t) \; - \; c_{k,t-1}(\alpha_t), \tag{10}$$

where $\hat{\mu}_{k,t-1}(\alpha_t)$ is the MU's estimate for the expected task latency $\mu_k(\alpha)$ and $c_{k,t-1}(\alpha_t)$ is an exploration term of BS k given context α_t. $\hat{\mu}_{k,t}(\alpha_t)$ and $c_{k,t}(\alpha_t)$ are calculated as

$$\hat{\mu}_{k,t}(\alpha_t) = \frac{1}{N_{k,t}(\alpha_t)} \sum_{n=t-\tau+1}^{t} T_{k,n}(\alpha_n) \cdot \mathbf{1}_{\{\hat{k}_n=k \,\wedge\, \alpha_n=\alpha_t\}} \tag{11}$$

and

$$c_{k,t}(\alpha_t) = \xi\sqrt{\frac{\log(t-\tau)}{N_{k,t}(\alpha_t)}}, \tag{12}$$

where $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function and $N_{k,t}(\alpha_t)$ is the number of times a BS k was selected in combination with context α_t. It is given by

$$N_{k,t}(\alpha_t) = \sum_{n=t-\tau+1}^{t} \mathbf{1}_{\{\hat{k}_n=k \,\wedge\, \alpha_n=\alpha_t\}}. \tag{13}$$

ξ is an algorithm parameter that tunes the exploration behavior of the MU, i.e., the larger ξ, the larger the exploration term and the more likely the MU is to explore other potentially suboptimal BSs. Moreover, τ is the last time step before t at which a change in a task latency distribution was detected.

[1]Under some regularity conditions for the context space, it can be quantized into a partition consisting of equally large hypercubes. The MU would then have to learn the expected reward for every hypercube-arm combination.

C. Change Point Detection

C-CPD-UCB adapts to non-stationarities by identifying change points in the task latency distributions. Motivated by [15], the intuition behind our proposed mechanism for the change point detection is to compare the $W \in \mathbb{N}$ most recent task latency samples $T_{\hat{k}_n,n}(\alpha_n),\ n = t-W+1,\dots,t$ with the long-term task latency estimates $\hat{\mu}_{\hat{k}_n,t}(\alpha_n),\ n = t-W+1,\dots,t$ that correspond to the last W selected BSs and task types. Compared to doing a comparison for every BS and task type combination separately, this allows for a quicker detection without the necessity to acquire many task latency samples per BS and task type pair. This is especially helpful, since, due to the large number of BS and task type pairs, there might be a scarcity of task latency samples for every combination. In particular, a change point in a task latency distribution is detected if the following inequality is satisfied

$$\frac{1}{W}\sum_{n=t-W+1}^{t} \mid T_{\hat{k}_n,n}(\alpha_n) - \hat{\mu}_{\hat{k}_n,t}(\alpha_n) \mid \geq \delta, \tag{14}$$

where $\delta > 0$ is a threshold parameter for the detection. Both, W and δ are input parameters of the algorithm, that tune the sensitivity for the detection of non-stationarities. In particular, the smaller W and δ, the higher the detection sensitivity. However, the likelihood of false detections also increases with smaller W and δ. In case a change in the task latency distributions is detected in time step t, the algorithm enters a reset phase. In order for Equation (14) to be satisfied, it is likely that multiple task latency samples are needed after the occurrence of a non-stationarity. Therefore, it is plausible to assume that the detection of a change point is always slightly delayed. Hence, τ is set to $t-W$ and the task latency estimates are updated based only on the last W task latency samples. This also accelerates the convergence after the detection, as the algorithm already acquired W samples for the estimation.

Algorithm 1 C-CPD-UCB

```
1:  Input Parameters: $T$, $W$, $\xi$ and $\delta$
2:  Set $N_{k,0}(\alpha) = 0,\ \forall k, \forall\alpha$.
3:  Set $\tau = 0$.
4:  for each $t = 1,\dots,T$ do
5:      Observe task type $\alpha_t$.
6:      if $N_{k,t-1}(\alpha_t) > 0\ \forall k \in \mathcal{K}$ then
7:          Select BS $\hat{k}_t$ according to (10).
8:      else
9:          Select BS $\hat{k}_t$ randomly from set $\{k \in \mathcal{K} | N_{k,t-1}(\alpha_t) = 0\}$.
10:     end if
11:     Observe delay $T_{\hat{k}_t,t}(\alpha_t)$ and update $\hat{\mu}_{\hat{k}_t,t}(\alpha_t)$, $N_{\hat{k}_t,t}(\alpha_t)$ and $c_{\hat{k}_t,t}(\alpha_t)$
        based on Equations (11), (13) and (12), respectively.
12:     if Equation (14) is satisfied AND $N_{k,t}(\alpha) > 0\ \forall k \in \mathcal{K}, \forall\alpha \in \mathcal{A}$ then
13:         Set $\tau = t - W$.
14:         for $k \in \mathcal{K}$ do
15:             for $\alpha \in \mathcal{A}$ do
16:                 Update $N_{k,t}(\alpha)$ according to Equation (13).
17:                 if $N_{k,t}(\alpha) > 0$ then
18:                     Update $\hat{\mu}_{k,t}(\alpha)$ and $c_{k,t}(\alpha)$ based on Equations (11) and (12),
                        respectively.
19:                 end if
20:             end for
21:         end for
22:     end if
23: end for
```

TABLE I
Simulation Parameters [4]

Parameter	Value	Parameter	Value
B	10 MHz	$\beta_t(1)$	100 MB
σ_n^2	10^{-13} W	$\beta_t(2)$	1 MB
P	200 mW	$\beta_t(3)$	10 MB
γ	3	$\beta_t(4)$	25 MB
ν_k	$[2, 6]$	λ_k	$[2, 10]$
κ_k	$[0, 5]$	$\eta_k(\alpha)$	$[0.2, 0.6]$

D. Regret

The performance and convergence behavior of an MAB algorithm are usually evaluated using the notion of regret. The regret is defined as the cumulative difference between the expected rewards obtained from selecting the optimal arms $k_t^* = \arg\min_{k \in \mathcal{K}} \mu_k(\alpha_t)$ and the actual obtained reward resulting from the algorithm selecting arms $\hat{k}_t$, i.e., $\mathcal{R}(T) = \sum_{t=1}^{T} T_{\hat{k}_t,t}(\alpha_t) - \mu_{k_t^*}(\alpha_t)$.

V. Numerical Results

For our simulations, we consider an area of 500 m $\times$ 500 m divided into a grid consisting of 16 equally sized squares. If not specified otherwise, we place $K = 9$ BSs uniformly on the inner grid nodes. The MU is placed randomly inside of the area and it is assumed that the MU only moves on a small scale. If not stated otherwise, $A = 4$ different task types are considered. Similar to [4], these task types correspond to different classes of applications with different task sizes as well as hardware and software requirements, i.e., video-processing, computation (e.g., MATLAB), audio and photo processing [4], or the processing of sensor data [8]. The task type is drawn randomly for every time step with equal probability. The simulation parameters are displayed in Table I.

In order to capture the dynamic non-stationary nature of the wireless channel as well as the BS load, it is assumed that the underlying probability distributions can change up to $\Gamma(T)$ times per simulated scenario, i.e., there are $\Gamma(T) + 1$ stationary phases. For every stationary phase, the values of the distribution parameters are randomly drawn from uniform distributions with intervals specified in Table I. As performance benchmarks, we consider the following MAB approaches from the literature. Note that all of the considered algorithms incorporate context, i.e., the task type.

- *C-UCB*: Contextual variant of the classic UCB algorithm [14] for stationary bandits.
- *C-ϵ-greedy*: This algorithm picks a BS randomly with probability ϵ^t, where $\epsilon \in (0, 1)$, and exploits the BS with the lowest estimated task latency with probability $1 - \epsilon^t$.
- *C-SW-UCB* and *C-D-UCB*: Contextual extensions of the non-stationary MAB algorithms Sliding-Window UCB and Discounted UCB proposed in [16].

The parameters for all the baseline algorithms are tuned such that they can achieve their best performance for the considered scenario. For our proposed algorithm, we chose $\xi = 0.012$, $\delta = 0.05$ and $W = 40$. For every result shown, $T = 2000$ time steps and the average of 200 Monte Carlo runs are considered. The regret is the metric used to assess the performance of our proposed C-CPD-UCB and the baseline algorithms. It reflects the performance loss, in terms of task latency, compared to selecting the optimal BS.

In Figure 2, the per time step regret $T_{\hat{k}_t,t}(\alpha_t) - \mu_{k_t^*}(\alpha_t)$ is displayed for a scenario with one distribution change point at $t = 1000$. Figure 2 illustrates the algorithms' behavior during stationary phases as well as the reaction time and convergence after the occurrence of a non-stationary event. The initial convergence behavior of all algorithms is similar, as they are capable of achieving a regret of less than one second within 100 time steps. After $t = 300$, our proposed C-CPD-UCB as well as the stationary MAB algorithms C-ϵ-greedy and C-UCB maintain their regret, indicating that they converge. However, C-D-UCB and C-SW-UCB, exhibit an increased regret during the first stationary phase, i.e., until $t = 1000$. This behaviour is caused by the mechanisms they use to adapt to non-stationary environments. In order to adapt to changes in the reward distribution, C-D-UCB applies a discount factor and C-SW-UCB applies a sliding window to the reward history. That means that the most recent reward samples have a greater influence on the expected reward estimates compared to older ones. In consequence, there are less samples available for an accurate estimation of the expected task latencies, which in turn causes more frequent exploration of suboptimal BSs. For C-D-UCB, this happens in a gradual manner, while for C-SW-UCB, these exploration phases come in bursts. Our proposed C-CPD-UCB does not suffer from this issue as it aims for actively detecting changes in the reward distributions rather than passively always reducing the influence of older reward samples. By tuning the detection parameters δ and W appropriately to avoid an overdetection, C-CPD-UCB can achieve the same performance as C-UCB during stationary phases. After the non-stationarity at $t = 1000$, C-UCB and C-ϵ-greedy show a slow reaction time and, on average, fail to converge back to the optimal solution. This is explained by C-UCB and C-ϵ-greedy using all past reward samples for the estimation of the expected task latency. That means that new additional samples do not contribute much after a large number of reward samples has been acquired. Although based on UCB, our proposed algorithm does not suffer from this limitation because it aims to actively detect change points in the reward distributions. Compared to C-D-UCB, our solution shows a similar reaction time. However, as discussed before, C-D-UCB shows a performance loss during the succeeding stationary phase. In comparison with C-SW-UCB, our proposed algorithm is capable of reacting to the distribution change point within half of the amount of time steps.

As explained in Section IV, extending existing solutions for non-stationary MABs, like SW-UCB and D-UCB, to handle context is not a trivial task. To support this hypothesis, Figure 3 displays the average cumulative regret $\mathcal{R}(T)/T$ for $\Gamma(T) = 2$ distribution change points versus the numbers of considered task types. The change points are located at random time steps. The more task types are considered, the more challenging the learning problem becomes, because there are more BS and task type combinations to learn. It can be seen that C-

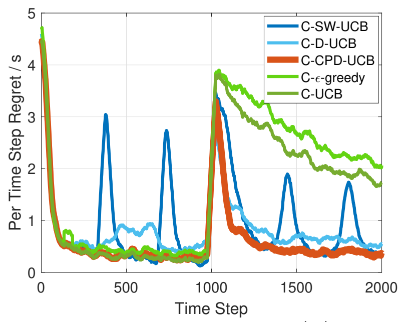

Fig. 2. Regret per time step for $\Gamma(T) = 1$ and $K = 9$ BSs.

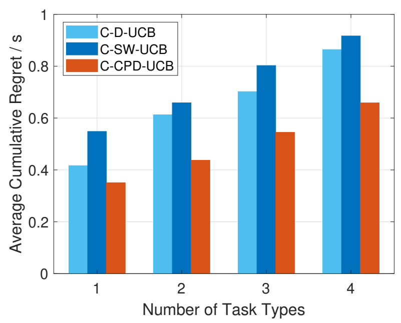

Fig. 3. Regret versus number of task types for $\Gamma(T) = 2$ and $K = 9$ BSs.

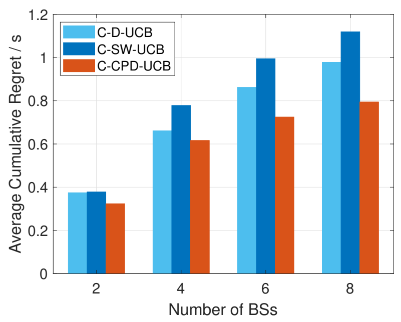

Fig. 4. Regret versus number of BSs for $\Gamma(T) = 2$.

SW-UCB and C-D-UCB scale poorly with increasing number of task types compared to our proposed algorithm. For C-D-UCB, the regret increases by 107% when going from one task type up to four, compared to an increase of only 88% for our proposed solution. This is due to the fact that for an increasing number of task types, there are less task latency samples for each BS and task type combination available. C-SW-UCB and C-D-UCB rely on putting more weight on the most recent task latency samples, which might be scarce for every BS-task type pair. This results in a poor performance during stationary phases due to a lack of samples for a reliable estimation and therefore, more frequent explorations.

Figure 4 illustrates the impact of the number of BSs K on the average cumulative regret $\mathcal{R}(T)/T$ for $\Gamma(T) = 2$. For Figure 4, we place the BSs uniformly and symmetrically on the inner grid nodes inside of the considered area. With increasing K, there are more BS and task type combinations to learn. Thus, the learning problem becomes more challenging. Similar to the behavior observed in Figure 3, with an increasing number of BSs K, the gains obtained by C-CPD-UCB compared to the baseline algorithms become more pronounced. Specifically, for $K = 2$ BSs, compared to our C-CPD-UCB, C-D-UCB and C-SW-UCB exhibit a 16% and 17% higher regret, respectively. For $K = 8$ BSs, this gap widens to 23% for C-D-UCB and 40% for C-SW-UCB, indicating that our proposed algorithm scales better with an increasing number of BSs.

VI. Conclusion

We considered latency minimization in MEC with heterogeneous edge servers. A MU sequentially selects the best BS for offloading computation tasks. Due to different hardware and software configurations, the BSs exhibit different computation performance, i.e., latency, depending on the type of computation task and its requirements. Furthermore, we considered a dynamic environment in which transmission rates to the BSs and the edge servers' task execution latencies are modelled with piece-wise stationary probability distributions. We proposed a piece-wise stationary contextual MAB algorithm that treats the task type as context and detects changes in the latency probability distributions. Simulations showed that our proposed algorithm exhibits excellent performance during stationary phases and a quick detection and adaptation in case a non-stationarity occurs.

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] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: integrating cloudlets and mobile edge computing," in *2016 23rd International conference on telecommunications (ICT)*. IEEE, 2016, pp. 1–5.

[4] T. Mahn and A. Klein, "Energy-efficient application-aware mobile edge computing with multiple access points," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2020, pp. 1–7.

[5] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[6] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *Ieee Access*, vol. 6, pp. 12 825–12 837, 2018.

[7] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 1468–1476.

[8] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4233–4248, 2021.

[9] S. Ghoorchian and S. Maghsudi, "Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 279–293, 2020.

[10] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing," *IEEE Access*, vol. 7, pp. 160 916–160 926, 2019.

[11] Y. Zhou, C. Shen, and M. van der Schaar, "A non-stationary online learning approach to mobility management," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1434–1446, 2019.

[12] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[13] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "A control and data plane split approach for partial offloading in mobile fog networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[14] R. Agrawal, "Sample mean based index policies by o (log n) regret for the multi-armed bandit problem," *Advances in Applied Probability*, vol. 27, no. 4, pp. 1054–1078, 1995.

[15] G. Ghatak, "Actively tracking the optimal arm in non-stationary environments with mandatory probing," *arXiv preprint arXiv:2205.10366*, 2022.

[16] A. Garivier and E. Moulines, "On upper-confidence bound policies for non-stationary bandit problems," *arXiv preprint arXiv:0805.3415*, 2008.