Towards user-centric, switching cost-aware fog node selection strategies

Zeineb Rejiba, Xavier Masip-Bruin, Eva Marín-Tordera

Advanced Network Architectures Lab (CRAAX) - Universitat Politècnica De Catalunya (UPC)

Abstract

In order to address high latency issues that may arise when executing time-critical applications at the cloud side, the novel fog computing paradigm has emerged, thus enabling the execution of such applications within computation nodes present at the edge of the network. While executing such applications, a user may be moving in an area where a high number of heterogeneous fog nodes (FNs) co-exist. This makes the problem of selecting the most appropriate fog node to execute the user's tasks challenging, especially since the set of visible FNs dynamically changes. Therefore, to deal with the uncertain and dynamic nature of such a fog computing environment, we model the FN selection problem using multi-armed bandits. However, standard solutions for the bandit problem are not tailored for scenarios with changing FN availabilities. In addition, since switching from one FN to the other causes a switching cost, such solutions lead to accumulating a high switching cost. Therefore, to address these issues, we first propose a block-based FN selection scheme, where switching among FNs is not allowed during a block of timeslots. We also propose a greedy approach, where FNs having a sufficiently good delay performance are selected in a greedy manner. Simulation results reveal that both approaches significantly improve the FN selection performance. In particular, we found that the block-based selection results in the lowest switching costs, whereas the greedy selection achieves the best overall performance.

[1]E-mails: {zeinebr,xmasip,eva}@ac.upc.edu

Keywords: fog computing, edge computing, mobility management, multi-armed bandits, switching cost

1. Introduction

During the past years, cloud computing has been considered as the standard for hosting different types of applications due to its flexibility and its pay-per-use model. However, as the quality of service (QoS) requirements of certain applications increase, high latencies resulting from remote cloud usage can no longer be tolerated. This is the case for example for augmented reality, virtual reality, online mobile games, etc. As a result, this has given rise to new architectural paradigms such as fog [1] or edge [2] computing, which move computation servers towards the edge of the network, typically one wireless hop away from end-users [3].

Depending on the considered standardization body, these computation nodes made available at the close vicinity of users are usually referred to as fog nodes (FNs) in the reference architecture by the OpenFog Consortium [4][2] or multi-access edge computing (MEC) servers, as defined by the European Telecommunications Standards Institute (ETSI) [5]. For instance, they may be co-located with wireless access points, cellular base stations, or have their dedicated hardware deployed by city managers, or even be contributed from idle user devices with sufficient capacity [6]. Fog nodes could also be seen as a logical entity created on top of different physical devices, thus allowing a distributed service execution [7]. We refer the reader to [7] for a more thorough discussion on this terminology and we will use the term *fog node* in the remainder of this article, except while exposing related works, where we will keep the original term used by the authors.

One of the problems that arise in this context is how to appropriately select an FN to execute a user's computation-intensive tasks. This problem is particu-

[2]The OpenFog Consortium was later merged with the Industrial Internet Consortium.

larly challenging when the considered fog computing environment is dense, with dynamically-changing FN availabilities due to user mobility. On the one hand, the mobility of the user in a high-density fog area will lead to accumulating a high switching cost from one FN to the other, if no suitable policy is implemented to control it. On the other hand, the dynamic availabilities of FNs, in addition to their fluctuating loads and capacities result in an uncertain environment, which brings an additional level of difficulty for finding the optimal FN.

As a result, to deal with this uncertainty, recent works have started to formulate the FN selection problem as a sequential decision-making problem, based on multi-armed bandits (MAB)- or Markov Decision Process (MDP)-based reinforcement learning (RL) approaches. In such RL approaches, a decision-making agent - placed either within the user's device, the fog nodes, or a separate control entity - interacts with an environment and selects actions (FNs) in a sequential manner. In the MAB case, the decisions are made based on the rewards returned by the environment, whereas in the MDP case, the environment's current state is also taken into account. The overall goal is to maximize the accumulated reward.

In this paper, we focus on the case where the decision-making logic is implemented in a software component in the user's device. Therefore, we focus on bandit approaches to provide a low complexity solution that can run on users' devices. In this case, the FNs would represent the actions to be selected by the agent and the quality of each FN is measured in terms of its task execution delay, which needs to be minimized (instead of maximizing the reward in the standard formulation).

Similar works that use bandits for FN selection can be found in the literature [8, 9, 10, 11, 12, 13, 14]. However, they either consider a stationary user or an indoor user mobility, where the set of FNs is fixed. In addition, they do not appropriately control the switching costs, which would lead to a negative impact on the overall task execution performance.

Therefore, to deal with the aforementioned issues, our aim in this paper is to propose user-centric FN selection strategies, with a specific focus on user mobility, high FN density and the switching costs that may be incurred in such a context.

We initially propose a Block-based FN Selection (BFS) approach, which is a variant of the standard Upper Confidence Bound (UCB) algorithm [15]. However, unlike a UCB-based approach where the selection of a different FN is allowed in consecutive timeslots, our proposed BFS approach forces the selected FN to remain the same during a number of consecutive timeslots (i.e. a block of timeslots), thus reducing the number of switches from one FN to another.

Although this approach provides an improvement in the performance as will be shown in Section 5, the exploration inherently performed in the UCB algorithm could lead to selecting a sub-optimal FN for a whole block of timeslots, which may lead to increased task delays. Then, to deal with this issue, we present a greedy alternative, termed Adaptive Greedy FN Selection (AGFS) approach. In this approach, if the agent has found an FN with a good enough delay estimate, it will select it in a greedy manner. Otherwise, it searches for FNs with a better quality (i.e. lower delays).

To evaluate the proposed approaches, extensive simulations have been conducted using a realistic mobility trace. The obtained results show that BFS and AGFS result in significant improvements in the average task execution delay and substantially reduce the accumulated switching costs. More specifically, BFS outperforms AGFS in terms of accumulated switching costs, due to its block-based nature, whereas AGFS outperforms BFS in the overall average delay.

To summarize, the contributions of our paper are as follows:

- Given the dynamic and uncertain nature of the considered fog computing environment, we model the FN selection problem as a multi-armed bandit problem with a delay minimization objective. Different delay components are considered, including a stochastic *waiting delay* that further contributes to the system's uncertainty.

- In order to reduce the costs associated with switching from one FN to another, we propose a block-based FN selection scheme that reduces the occurrence of switching by keeping the same FN selected during a block of timeslots. This FN will be selected by minimizing the upper confidence bound over the set of visible FNs.

- We also propose an adaptive greedy approach, where the FN having the minimum average delay estimate is selected, if such an estimate is good enough. Otherwise, the user is allowed to perform an exploration step to find FNs offering lower delays.

- Both approaches are evaluated based on a real access point availability dataset, in order to model different levels of FN deployment densities.

The remainder of this paper is organized as follows: In Section 2, we provide an overview of the related works. Section 3 defines the system model and describes the problem to be addressed. Section 4 presents a review of the MAB theory and then describes BFS and AGFS, our two proposed solutions for FN selection. The simulation setup and the obtained results are analyzed in Section 5. Finally, we conclude the paper in Section 6.

2. Related works

As stated in Section 1, as more and more emphasis is being placed on moving intelligence to end user devices [16], we focus on the case where the FN selection strategy is implemented within a software component at the user side. Related works in this category include [8] where the authors propose a MAB-based strategy to select the best base station (BS) to execute a user's task in a dense edge computing environment. The authors consider a variable number of BSs in addition to considering a switching cost when different BSs are selected in consecutive timeslots. However, they do not provide a mechanism to control this cost effectively. In [9], a similar approach is considered for FN selection, both when the delay distributions of each FN are stationary and non-stationary.

Similar to our work, the authors consider that FNs' arrivals and departures are unpredictable, however they do not consider switching costs. In [10], the authors consider the problem of how a task node selects a helper node in a fog computing context. Given that helper nodes have unknown delay distributions, the MAB approach is adopted, with a specific focus on the non-stationarity of the delay distributions. In [11], the authors present a MAB-based strategy allowing a user to select the best neighboring fog node to transmit its tasks to. This work focuses on the case where FNs have heterogeneous preferences towards task types and where the user receives a binary feedback indicating whether the FN is satisfied with the type of the task assigned to it or not. The authors in [12] consider the scenario where a user has to select an edge computing server with the highest probability of meeting the task deadline. More specifically, the user has to deal with the constraints placed on the energy budget in addition to the non-stationarity of the deadline violation probabilities. In a previous work [13], we considered the FN selection problem in a high density fog computing environment. Since the user's mobility along with the high FN density does not leave the user enough time to learn the delay performances of all nearby FNs, only a limited subset of FNs was considered in our proposed MAB strategy in order to improve the learning performance. Authors in [14] use a contextual MAB approach to determine the best service placement among a set of edge servers, the user's device and cloud servers. The aim is to minimize the weighted sum of the computation delay, the communication delay and the switching cost. However, their context representation is tailored for the case of a fixed set of edge servers and as a result would not be well-suited for the case where the set of servers dynamically changes.

Other than MAB-based approaches, other reinforcement learning techniques have been used in the literature for the problem of FN/MEC server selection. In [17], the authors present a Q-learning approach that takes into account the current serving BS and the channel state in order to select the target BS with the objective of minimizing the task execution speed. In [18], the authors propose the use of a double deep Q-network to allow a device to intelligently decide among local task execution or selecting a BS through which the task will be offloaded.

Table 1: Comparison of related works

Reference	Approach	Variable number of FNs /MEC servers	Switching cost considered
[8]	MAB	✓	✓(but not controlled)
[9]	MAB	✓	✗
[10]	MAB	✗	✓
[12]	MAB	✗	✗
[11]	MAB	✗	✗
[13]	MAB	✓	✓(but not controlled)
[14]	Contextual MAB	✗	✓
[17]	MDP + Q-learning	✗	✓
[18]	MDP + Double deep Q-network	✗	✓

Table 1 summarizes the differences between the aforementioned related works. As it can be seen, there is a lack of works that efficiently handle the dynamically-changing FN availabilities and the presence of switching costs, which is why we aim to jointly address these issues in this paper.

3. System model

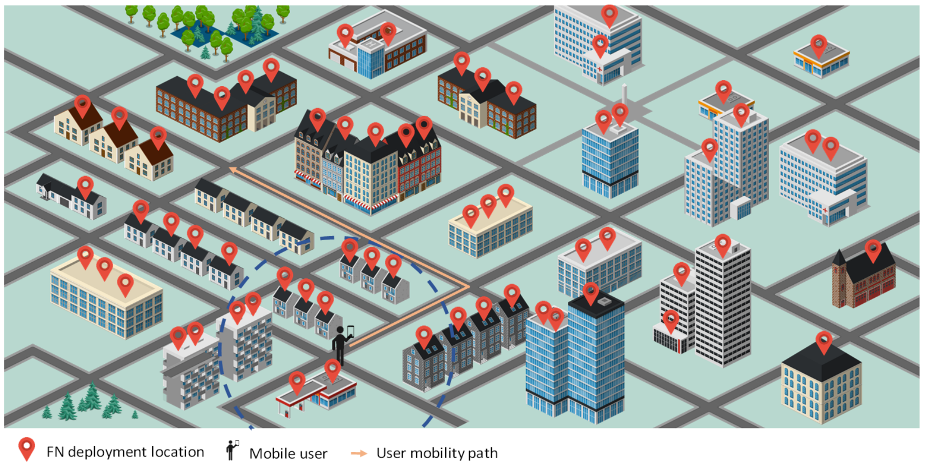

Figure 1: FN deployment scenario

In this work, we consider a fog computing scenario where fog nodes are densely deployed in an urban environment, as shown in Fig. 1. We focus on the case of a representative user moving in this environment and running an application that generates a sequence of independent computation-intensive tasks $\{T_k\}_{k=1}^{K}$ that need to be executed by one of the fog nodes available in the user's vicinity. We denote by $\mathcal{A}_k = \{a_i, i \in [1, |\mathcal{A}_k|]\}$ the set of FNs available at the time in which task T_k was generated. This set periodically changes every Δ seconds due to user mobility.

Each task T_k is characterized with its input data size α_k, its computation intensity ω_k (in terms of number of required CPU cycles per input bit) and its output data size β_k. When an FN is selected for processing a task, a delay will be incurred. It is comprised of the following components:

- Transmission delay d_{tx}^{k}: It is the time needed to transmit the task input data on the wireless channel. It can be determined as follows:

$$d_{tx}^{k} = \frac{\alpha_k}{R_i} \tag{1}$$

where R_i is the uplink transmission rate between the user and the selected FN a_i. It is derived as follows:

$$R_i = W \log\left(1 + \frac{P_{tx} H}{\sigma^2 + I}\right) \tag{2}$$

where W is the wireless channel bandwidth, P_{tx} is the transmit power of the user's device, σ^2 is the noise power, I is the interference power caused by other users connected to the same FN, and H is the path loss. To model the path loss, we adopt the IEEE TGn task group channel model [19]:

$$H_{dB}(d_i) = \begin{cases} H_{FS}(d_i)\ , d_i \leq d_0 \\ H_{FS}(d_0) + 35\log_{10}(\frac{d_i}{d_0})\ , d_i > d_0 \end{cases} \tag{3}$$

where d_i is the user-FN distance, H_{FS} is the free space path loss and d_0 is the reference distance. As we are considering pedestrian mobility in an urban environment, we specifically adopt model F in [19] where $d_0 = 30m$.

- Waiting delay d_w^k: It models the delay experienced by the task T_k in the FN's task queue. We model it as a random variable whose distribution is not known at the user side a-priori[3]. In fact, several factors could cause a variation in d_w^k, such as the task arrival rate at that specific FN, the intensity of those tasks as well as the rate at which it is processing those tasks.

- Processing delay d_p^k: It is the time needed by the FN to process the task.

$$d_p^k = \frac{\alpha_k \omega_k}{f_i} \tag{4}$$

  where f_i is the CPU clock speed of the FN.

- Result transmission delay d_r^k: It is the time needed to return the computation results to the user. As is common in related literature [8, 17], this delay is often negligible, since the size of the result is small compared to the input size.

- Switching delay d_{sw}^k: A switching delay will be incurred whenever the FN selected to execute task T_k is different from the one selected to execute the previous task T_{k-1}. This could correspond to the delay needed to re-associate with the new FN, and the time needed by the latter to instantiate a new container to run the user's tasks.

$$d_{sw}^k = \begin{cases} C & a_k \neq a_{k-1} \\ 0 & a_k = a_{k-1} \end{cases} \tag{5}$$

Taking into account the aforementioned delay components, the total delay experienced by task T_k can be then expressed as:

$$d^k = d_{tx}^k + d_w^k + d_p^k + d_{sw}^k \tag{6}$$

[3]In the simulations in Section 5, we model it using a normal distribution.

Our goal is then to derive an FN selection strategy that minimizes the accumulated delay for the sequence of tasks $\{T_k\}_{k=1}^{K}$:

$$\min_{a_i \in \mathcal{A}_k} \sum_{k=1}^{K} d^k \tag{7}$$

Since multiple components of the delay are stochastic and are not available at the user side a-priori, the FN selection problem falls within the framework of sequential decision-making under uncertainty. More specifically, we will use multi-armed bandits to achieve the objective in (7), as we describe in the following section.

4. Proposed FN selection strategies

In this section, we first provide an overview of the multi-armed bandit framework and how it can be mapped to our context. We then present our algorithms for FN selection based on this framework.

4.1. Introduction to multi-armed bandits

In multi-armed bandits, a decision-making agent has to select an action among a set of actions $\mathcal{A}$. Each action a is characterized with a different reward distribution with mean μ_a, which is unknown to the decision maker a-priori. The goal of the decision maker is to maximize its accumulated reward over a given decision-making horizon T.

However, since the reward of each action is not known and is only revealed *after* the action is selected, the agent faces an exploration-exploitation dilemma. On the one hand, it has to explore different actions in order to create an accurate estimate of their rewards, thus potentially incurring short term reward losses if the explored actions are suboptimal. On the other hand, it has to exploit the action currently having the maximum reward estimate in order to increase the accumulated reward.

In order to measure the performance of a MAB agent, it is common to consider the expected regret[4], which measures the difference between the rewards achieved by an oracle having exact prior knowledge about the rewards of each action and rewards obtained by the considered action selection strategy:

$$\mathbb{E}[R_n] = \mathbb{E}[\sum_{i=1}^{n}(\mu^* - \mu_i)] \tag{8}$$

where R_n is the regret at timeslot n, μ^* is the mean reward of the best action selected by the oracle and μ_i is the mean reward of the action selected at timeslot i.

In order to appropriately select actions in a bandit problem while addressing the exploration-exploitation tradeoff, the Upper Confidence Bound (UCB) [15] algorithm is commonly used since it is guaranteed to converge to the optimal action. In a nutshell, UCB creates a confidence bound with respect to the current estimate of the reward of each action as follows:

$$i_a = \hat{\mu}(a) + \sqrt{\frac{\log(2t)}{N_t(a)}} \tag{9}$$

where $\hat{\mu}(a)$ is the estimate of the mean reward of action a, and the term on the right hand side is the confidence bound (also called the *exploration bonus*), t is the current timeslot and $N_t(a)$ is the number of times in which action a was selected up to time t.

It then selects the action with the maximum value of i_a. As it can be seen, the presence of $N_t(a)$ in the denominator implies that selecting an action more often will increase the certainty about its reward (which was initially unknown) and therefore result in refining its bound.

Following this framework, our FN selection problem can be formulated as a MAB problem, where the decision-making entity in the user's device corresponds to the agent, the actions to be selected are the fog nodes, and the delay taken

[4]Usually referred to as sampling regret.

by an FN to execute a user's task determines its quality compared to other FNs, and its sum has to be minimized.

However, the application of standard MAB and in particular the UCB algorithm, would be challenging in our scenario given the following considerations:

1. While in the standard MAB setting the set of actions $\mathcal{A}$ is fixed during the whole decision-making time horizon, in our context, the set of actions (i.e. FNs) dynamically changes as a result of the user's mobility. So, we only have a limited amount of time to learn the delay estimates of a high number of FNs, since we are considering a high-density scenario.
2. In our scenario, a switching cost is incurred whenever an action is changed compared to the previous one, as opposed to the standard MAB where such a cost does not exist. In this case, if we apply an approach such as UCB as it is, a high switching cost would be incurred, since it will keep switching among actions in order to explore them and refine their confidence bounds. The incurred loss is measured in terms of a switching regret [20], defined as:

$$R_{sw}(n) = C \sum_{i=2}^{n} \mathbb{1}\{a_i \neq a_{i-1}\} \tag{10}$$

   where C is a given switching cost.

   Given the aforementioned considerations, standard MAB algorithms should be carefully adapted to our setting and the algorithms presented in the next section will attempt to achieve this goal.

4.2. Block-based FN selection scheme (BFS)

An intuitive approach to avoid accumulating switching costs is to explicitly prevent switching during a given amount of time. One way to achieve that is the Block Allocation Scheme (BAS) presented in [20], which was among the first works to address a bandit problem with switching costs. The key idea in BAS was to select the same action during a given "block" of timeslots.

Therefore, our Block-based FN Selection (BFS), shown in Alg. 1 uses the block-based structure of BAS in conjunction with UCB. At the beginning, we

Algorithm 1: BFS

```
1  $b \leftarrow 1$
2  remaining slots to use the current block size $r \leftarrow 0$
3  **for** $k \leftarrow 1$ **to** $K$ **do**
4      **if** *$k$ is a timeslot in which the FN set changes* **then**
5          retrieve set of visible FNs $\mathcal{A}_k$
6          calculate $L$ according to Eq. 11
7      **if** $r = 0$ **then**
8          $a_{min} \leftarrow \arg\min_{a_i \in \mathcal{A}_k} \left( \frac{z(a_i)}{n(a_i)} - \sqrt{\frac{\log(2k)}{N_k(a_k)}} \right)$
9          $nb_{used} \leftarrow nb_{used} + 1$
10         **if** $nb_{used} > L$ **then**
11             $b \leftarrow b + 1$
12             $nb_{used} \leftarrow 0$
13         $r \leftarrow b$
14     $a_k \leftarrow a_{min}$
15     Observe delay $d(a_k)$
16     $z(a_k) \leftarrow z(a_k) + normalize(d(a_k) - d_{sw})$
17     $n(a_k) \leftarrow n(a_k) + 1$
18     $r \leftarrow r - 1$
19 **end**
```

start with a block size $b = 1$, which is equivalent to standard UCB. Later on, the block size is increased gradually depending on L, which represents the number of times in which the current block size should be kept the same. As suggested in [20], it is determined as follows:

$$L = \left\lceil \frac{2^{b^2} - 2^{(b-1)^2}}{b} \right\rceil . |\mathcal{A}_k| \tag{11}$$

where $|\mathcal{A}_k|$ is the current number of FNs.

After calculating L in Line 6, we check if an entire block has elapsed, in which case we proceed to selecting the FN according to UCB (Line 8). We increment the number of times in which the block size b has been used (Line 9). If this number exceeds L, then the block size should be increased by 1 (Line 11). We then reset the number of times where this new block size has been used and set the remaining timeslots to use this block size to the new b.

If the current block b has not entirely elapsed, we do not change the FN and select the one that has been determined previously. Next, the delay taken by the FN to execute the task is observed. Since the switching cost does not indicate the quality level of a given FN, it is subtracted in Line 16[5]. In Line 16, we also note that the resulting delay is normalized to the range $[0, 1]$ as it is necessary for UCB to operate in such a range. Finally, the number of times action a_k is selected is incremented while the number of remaining timeslots to use the current block size is decremented.

4.3. Adaptive greedy FN selection (AGFS)

As explained in the previous section, the block-based structure of BFS reduces the exploration frequency of UCB as well as the resulting accumulated switching cost. However, following this approach, when an FN is selected because its exploration bonus is high, it is possible that it is a suboptimal FN. As a result, the user remains connected to it for the whole block duration, which will severely affect the performance.

This leads us to propose a non UCB-based strategy. More specifically, we adopt an adaptive greedy approach that has been proposed in the context of mortal multi-armed bandits [21], which is a bandit variant where actions have a stochastic lifetime after which they become unavailable. This is similar to FNs in our case, which become unavailable as soon as the user leaves their coverage area. In this case, as stated in [21], finding a reasonably good action (FN) suffices and it is not necessary to select all actions indefinitely. This has a

[5]We assume the agent can estimate the switching cost by assessing the delay difference for the same FN in the case where a switch occurred and the case where a switch did not occur.

Algorithm 2: AGFS

```
1  for $k \leftarrow 1$ to $K$ do
2      if $k$ is a timeslot in which the FN set changes then
3          retrieve set of visible FNs $\mathcal{A}_k$
4      $a_{min} \leftarrow \arg\min_{a_i \in \mathcal{A}_k} \frac{z(a_i)}{n(a_i)}$
5      $p_{min} \leftarrow \frac{z(a_{min})}{n(a_{min})}$
6      draw a random number $n_r$
7      if $n_r \leq \min(1, c.(1 - p_{min}))$ then
8          $a_k \leftarrow a_{min}$
9      else
10         $a_k \leftarrow uniform(\mathcal{A}_k)$
11     end
12     Observe delay $d(a_k)$
13     $z(a_k) \leftarrow z(a_k) + normalize(d(a_k) - d_{sw})$
14     $n(a_k) \leftarrow n(a_k) + 1$
15 end
```

clear advantage when switching costs are present, since the same action will be selected in consecutive timeslots, which will naturally decrease the accumulated switching costs.

The pseudo-code for AGFS is depicted in Alg. 2. In each timeslot k, AGFS checks whether k is a timeslot in which new FNs become available. In this case, it gets the new list of FNs. Otherwise, it makes subsequent decisions based on the previous set of FNs.

Then, it determines the FN a_{min} with the lowest average delay estimate p_{min}[6]. Based on this, with probability $min(1, c.(1 - p_{min}))$, it selects a_{min} in a greedy manner. More in detail, when $1 \leq c.(1 - p_{min})$, it considers the

[6]Similar to BFS, AGFS also uses delays normalized in $[0, 1]$ in order to be able to transform them into probabilities.

performance of a_{min} sufficient and selects it with probability 1. Otherwise, it selects it with a lower probability $c.(1 - p_{min})$.

For example when $c = 1$ and the current best delay estimate is $p_{min} = 0.05$, the algorithm will select a_{min} with probability 0.95. More generally, c is a control parameter that determines the threshold for considering whether an FN is sufficiently good or not. A larger c results in an algorithm that considers FNs with a high delay estimate as "good enough" and exploits them greedily with probability 1. This should be avoided in situations where there exists an optimal FN with a delay ≈ 0. This is not the case in our scenario. That is why, in our simulations, a value of $c = 2$ has been considered and found to yield the best results.

If no FN has a satisfying average delay estimate, an exploration step will be performed by randomly selecting an FN among the visible ones, with the aim to find a better FN (Line 10).

After the FN is selected, its total delay is observed. Similar to BFS, the sum of delays of the selected FN is updated, along with the number of times it has been selected.

4.4. Complexity analysis

The computation complexity characterizing both BFS and AGFS originates from the calculation of a minimum value over the set $\mathcal{A}_k$ of visible fog nodes at timeslot k. This corresponds to Line 8 in Alg. 1 and Line 4 in Alg. 2, respectively. Since finding the minimum in a set of items of size A has a linear time complexity $\mathcal{O}(A)$, and since this step is repeated for K timeslots in both algorithms, the overall complexity is $\mathcal{O}(AK)$.

This result is similar to the other bandit algorithms that we use as benchmarks in the simulations in Section 5, since they also include this minimization step in each timeslot. As a result, the advantages of our proposed schemes are instead shown in terms of reductions in cumulative delays, which can be achieved thanks to the efficient handling of the dynamic availability of FNs, especially in presence of switching costs.

5. Simulation results

In this section, we outline the setup as well as the different parameters considered in our simulations. We then provide an analysis of the obtained results.

5.1. Simulation setup

In order to evaluate BFS and AGFS, we conducted a set of simulations in Python, using the models defined in Section 3 and the parameters in Table 2.

Table 2: Simulation parameters

Parameter	Value
Input data size α_k	1 $Mbit$
Computation intensity ω_k	2640 cycles per input bit
Channel bandwidth W	20 Mhz
Distance d_i	Uniformly at random in $\{10, 15, 20, 25, 30, 35, 40\}m$
Transmit power P_{tx}	0.5 W
Noise power σ^2	2.10^{-13} W
CPU clock speed f_i	$\{2, 3, 4.5\}$ GHz
Waiting delay d_w (in s)	$\mathcal{N}(\mu, \sigma)$, where $\mu = \mathcal{U}(0, 1)$ and $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$
Switching cost C	$\{50, 100, 200\}$ ms
c (in Alg.2)	2
Number of periods Δ where the set of FNs changes	64
Number of timeslots in each period	60
Total number of timeslots K	$64 \times 60 = 3840$

To model the dynamically-changing FN availabilities as the user moves, we used the mobility trace in [13]. In fact, the trace contains logs of the Wi-Fi access points (APs) detected by a mobile user's smartphone, as it moves at walking speed in the city.

A subset of the detected access points is then used to represent FNs in our fog computing scenario. This subset is formed as follows: we take all the APs having an SSID indicating a public Wi-Fi network, in addition to half of the

APs having an SSID indicating a private Wi-Fi network. This could correspond to a scenario where all public network owners have upgraded their APs as FNs, whereas only 50% of the private network owners have upgraded their APs into FNs. The resulting FN distribution is shown in Appendix A.

We compare BFS and AGFS to the following approaches:

- Auer [22]: It is a variant of UCB where the set of actions (FNs in our case) is time-varying. More specifically, it explores every new FN as soon as it appears and then selects FNs based on their confidence bounds.

- VUCB [23]: Similar to Auer, the set of actions in VUCB is time-varying. However, the difference is that the confidence bound of each action is adjusted based on its appearance time. This has been used for example in [8].

- Limexp: This approach has been proposed in our previous work [13]. It consists in running the UCB algorithm on a randomly-selected subset from the available FNs in each round. We run it with subset size $N = 4$, which was our best result in [13].

- Oracle: This approach represents the ideal, but hypothetical, scenario where the decision maker has access to the full system information before making its decisions. As a result, it will always choose the FN that will yield the lowest delay, and only switches to a different FN if a better FN appears during user mobility.

The following results were averaged over 50 runs.

5.2. Obtained results

5.2.1. Overall evaluation

Fig. 2 depicts the average cumulative delay (including the switching delay) of the different considered approaches. We first note that both proposed approaches significantly reduce this average delay compared to baseline approaches, including our previous work Limexp [13]. This is due to the fact

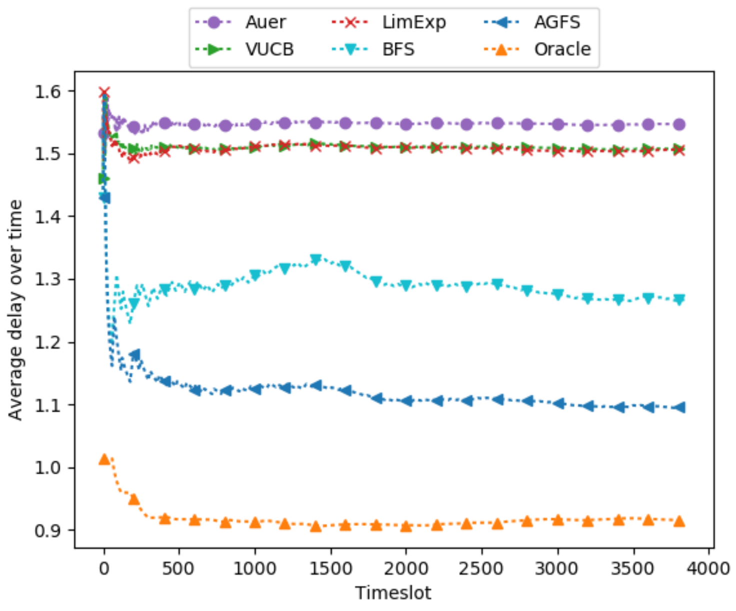

Figure 2: Average cumulative delay

that other approaches make use of UCB, which spends the limited FN availability time exploring the different FNs. Thus, it does not have enough time to settle on the optimal one. Even though the FN appearance time is taken into account in VUCB and a slight improvement is observed compared to Auer, it still results in a high delay. To avoid the afore-mentioned issue, BFS adopts the block-based selection, which prevents switching from one FN to another during the block duration. However, it is outperformed by the non UCB-based greedy AGFS approach.

In Fig. 3, we provide a closer look at the switching costs that have been accumulated by each approach. This corresponds to the switching regret defined in Eq. 10. As it can be seen, the accumulated switching costs of Auer, VUCB and Limexp rapidly grow as the number of timeslots increases. In contrast, both BFS and AGFS substantially reduce the switching costs . More specifically, since the frequency of switching is reduced by the block-based scheme in BFS, this results in an accumulated switching delay that is lower than that achieved by

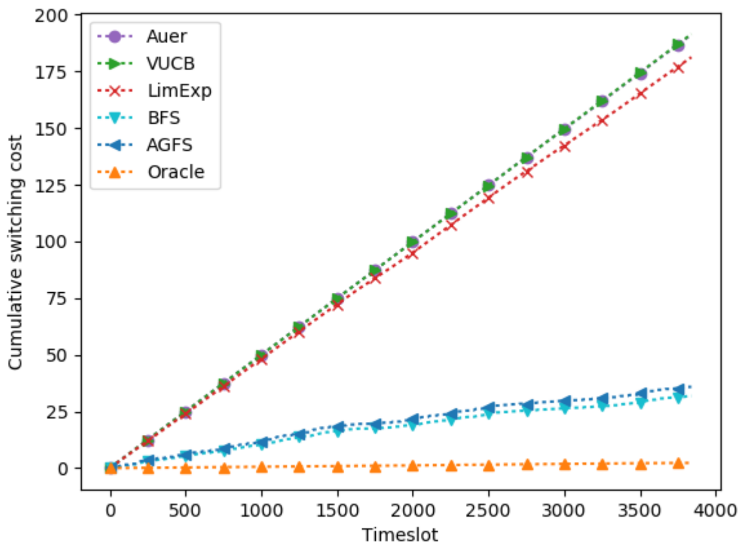

Figure 3: Cumulative switching cost

AGFS. We also note that the Oracle approach achieves a non-zero switching cost since it occasionally switches to a better FN, as soon as such an FN becomes available during the user mobility.

Fig. 4 shows the performance of each approach in terms of total regret, which is the sum of the sampling regret and the switching regret. AGFS achieves the lowest regret, which is followed by BFS, while Auer, VUCB and Limexp achieve a high total regret.

To further explain why such improvements have been obtained by BFS and AGFS, in Fig. 5, we plot the ratio of the simulation time that the user spent connected to the optimal FN. As it can be seen, among all algorithms, AGFS achieves the highest ratio due to its greedy nature. BFS, on the other hand, achieves a lower ratio, which is attributed to the fact that the block-based selection may cause the user to remain connected to a suboptimal FN for the whole block duration (i.e. for b timeslots). When this happens multiple times, this leads to a reduced ratio.

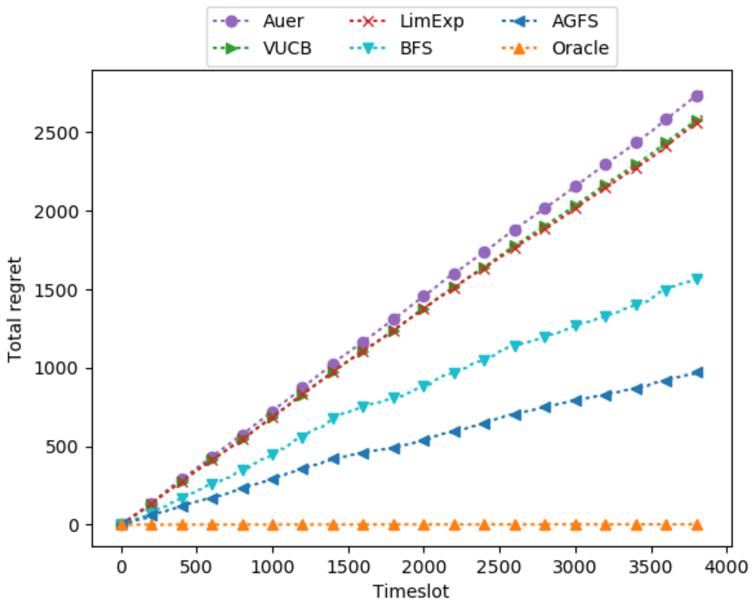

Figure 4: Total regret

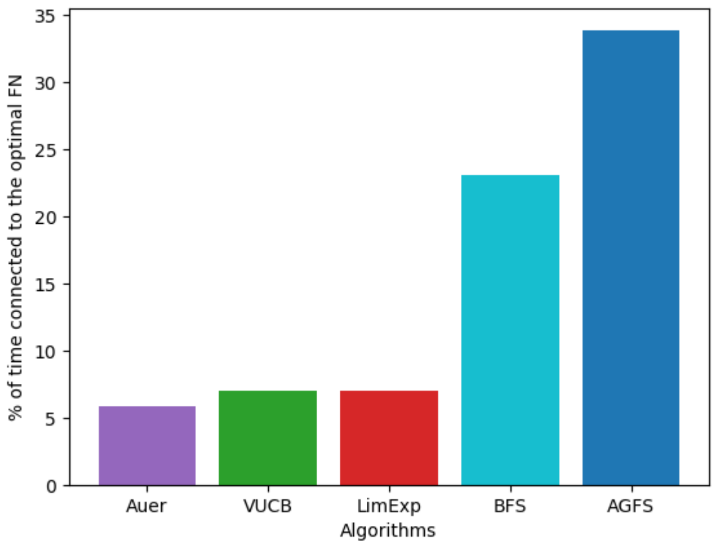

Figure 5: Ratio of simulation time where the user has selected the optimal FN

5.2.2. *Impact of the switching cost*

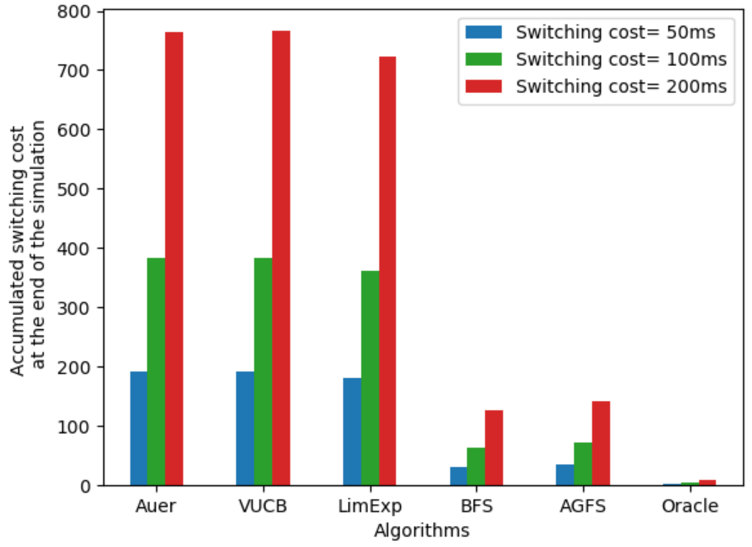

Figure 6: Accumulated switching cost

In Fig. 6, we analyze the impact of the switching cost on the different approaches. First, we note that the same pattern as Fig. 3 is observed, with AGFS obtaining a slightly higher switching cost than BFS. In addition, as the switching cost C increases, both BFS and AGFS accumulate substantially lower costs compared to Auer, VUCB and Limexp, which will positively impact the user experience.

5.2.3. *Impact of the FN density*

In this section, we study the impact of the FN density on the performance of the different algorithms. To this end, we use a different subset of the AP trace to represent FNs. Similar to the previous case, we consider all APs with an SSID indicating a public Wi-Fi network. Then, in contrast to the previous case, we consider a higher percentage, equal to 75%, of APs indicating private networks. The reader may refer to Appendix B for more details on the resulting

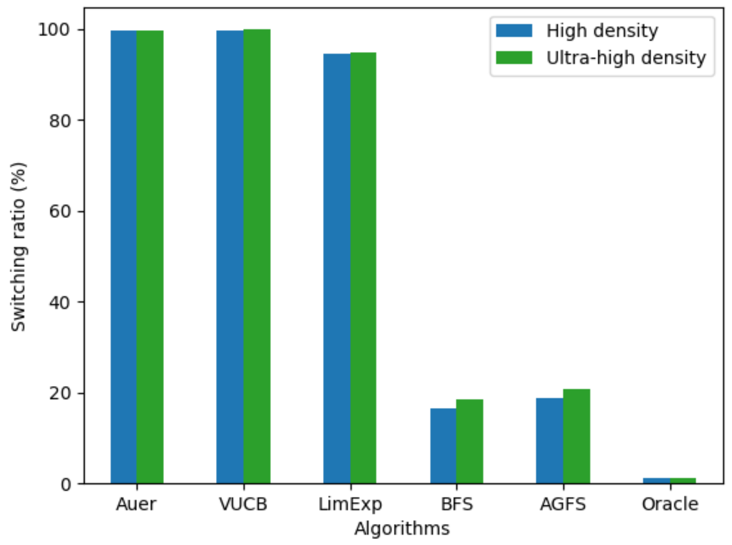

Figure 7: Switching ratio

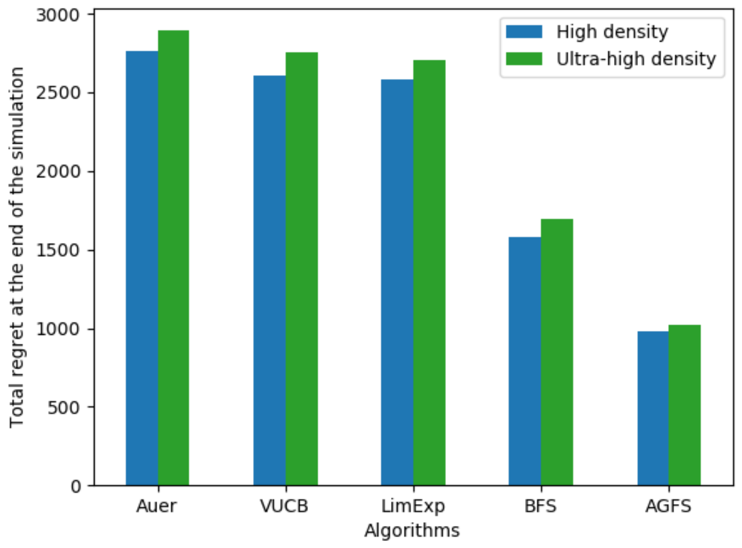

Figure 8: Comparison of the total regret for the two density cases

FN distribution.

We first plot the switching ratio in Fig. 7. It corresponds to the ratio of the number of timeslots in which a switching from one FN to another has occurred, out of the total number of timeslots. We note that in Auer, VUCB and Limexp, the increased FN density does not impact the switching ratio as it is already high. However, in BFS and AGFS, the increased FN density leads to an increased switching ratio, which is explained by the higher number of considered FNs.

In Fig. 8, we show the total cumulative regret obtained at the last timeslot of the simulation. As it can be seen, all approaches exhibit a similar behaviour, i.e. a higher FN density leads to a higher regret, which is mostly due to the increased contribution of the switching regret.

6. Conclusion

In this paper, we addressed the problem of user-centric fog node selection, with a specific focus on scenarios characterized with user mobility and a high density of FNs. The objective was to select the FN with the lowest, yet initially unknown delay while ensuring minimal accumulated switching costs from one FN to another. As this task involves sequential decision making in a dynamic fog environment characterized with uncertainty, we used the multi-armed bandits framework to address the problem. In fact, we first proposed BFS, a block-based FN selection scheme that keeps selecting the same FN during an entire block of timeslots. The second, AGFS, is an adaptive greedy approach, where, if a FN with a good enough delay estimate is found, it will be selected in a greedy manner. Simulation results show that both approaches significantly reduce the average task delay compared to baseline approaches that do not efficiently control the switching costs. In addition, while BFS can lead to a lower accumulated switching cost, AGFS can achieve a lower average delay. So, overall, we can note that it pays off to act greedily in scenarios with dynamic FN availabilities and when switching from one FN to the other causes a switching delay.

As a future work, we will investigate the possibility of incorporating context information in our bandit formulation and how this might affect the complexity of our solution. In addition, a multi-user scenario may be considered to examine the performance of the collective user behaviour in terms of FN selections.

Appendix A. Distribution of FNs in the high density scenario

Fig. A.9 shows the distributions of the FNs considered in the simulations, including the ones which are new compared to the ones that appeared during the previous period Δ.

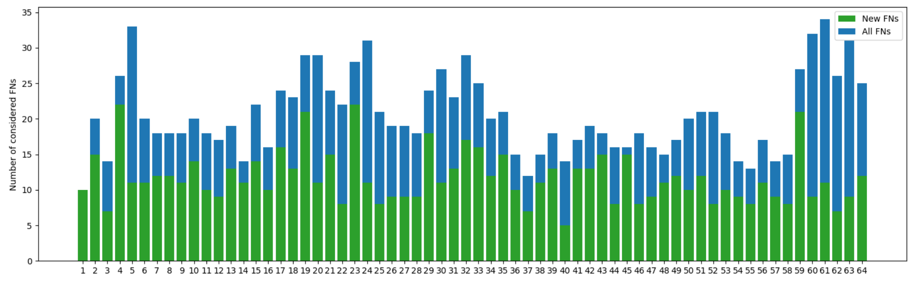

Figure A.9: Distribution of the FNs in the high density case

Appendix B. Distribution of FNs in the ultra-high density scenario

Fig. B.10 shows the distributions of the FNs considered in the ultra high density case in Section 5.2.3.

References

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

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE internet of things journal 3 (5) (2016) 637–646.

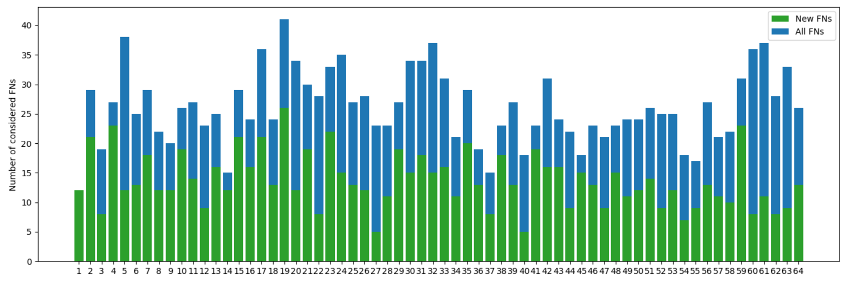

Figure B.10: Distribution of the FNs in the ultra-high density case

[3] M. Aazam, S. Zeadally, K. A. Harras, Fog computing architecture, evaluation, and future research directions, IEEE Communications Magazine 56 (5) (2018) 46–52.

[4] Openfog reference architecture, `https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf`, accessed: 2020-09-27.

[5] The european telecommunications standards institute, `https://www.etsi.org/technologies/multi-access-edge-computing`, accessed: 2020-09-27.

[6] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, G.-J. Ren, Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems, IEEE Wireless Communications 23 (5) (2016) 120–128.

[7] E. Marín-Tordera, X. Masip-Bruin, J. García-Almiñana, A. Jukan, G.-J. Ren, J. Zhu, Do we all really know what a fog node is? current trends towards an open definition, Computer Communications 109 (2017) 117–130.

[8] Y. Sun, S. Zhou, J. Xu, Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks, IEEE Journal on Selected Areas in Communications 35 (11) (2017) 2637–2646.

[9] Y. Tan, K. Wang, Y. Yang, M.-T. Zhou, Delay-optimal task offloading for dynamic fog networks, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.

[10] Z. Zhu, T. Liu, Y. Yang, X. Luo, Blot: Bandit learning-based offloading of tasks in fog-enabled networks, IEEE Transactions on Parallel and Distributed Systems 30 (12) (2019) 2636–2649.

[11] S. Zhao, Z. Zhu, F. Yang, X. Luo, Online optimal task offloading with one-bit feedback, in: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2018, pp. 683–687.

[12] S. Ghoorchian, S. Maghsudi, Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty, arXiv preprint arXiv:1904.06258.

[13] Z. Rejiba, X. Masip-Bruin, E. Marin-Tordera, A user-centric mobility management scheme for high-density fog computing deployments, in: 2019 28th International Conference on Computer Communication and Networks (ICCCN), IEEE, 2019, pp. 1–8.

[14] T. Ouyang, R. Li, X. Chen, Z. Zhou, 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.

[15] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine learning 47 (2-3) (2002) 235–256.

[16] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, P. Popovski, Five disruptive technology directions for 5g, IEEE Communications Magazine 52 (2) (2014) 74–80.

[17] J. Wang, K. Liu, M. Ni, J. Pan, Learning based mobility management under uncertainties for mobile edge computing, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–6.

[18] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, M. Bennis, Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning, IEEE Internet of Things Journal 6 (3) (2018) 4005–4018.

[19] V. Erceg, L. Schumacher, P. Kyritsi, et al., Ieee 802.11-03/940r4 tgn channel models, IEEE P802 11.

[20] R. Agrawal, M. Hedge, D. Teneketzis, Asymptotically efficient adaptive allocation rules for the multiarmed bandit problem with switching cost, IEEE Transactions on Automatic Control 33 (10) (1988) 899–906.

[21] D. Chakrabarti, R. Kumar, F. Radlinski, E. Upfal, Mortal multi-armed bandits, in: Advances in neural information processing systems, 2009, pp. 273–280.

[22] R. Kleinberg, A. Niculescu-Mizil, Y. Sharma, Regret bounds for sleeping experts and bandits, Machine learning 80 (2-3) (2010) 245–272.

[23] Z. Bnaya, R. Puzis, R. Stern, A. Felner, Social network search as a volatile multi-armed bandit problem, HUMAN 2 (2) (2013) 84.