EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks

Yuxuan Sun, Sheng Zhou, *Member, IEEE,* Jie Xu, *Member, IEEE*

***Abstract*—Merging mobile edge computing (MEC) functionality with the dense deployment of base stations (BSs) provides enormous benefits such as a real proximity, ultra-low latency access to computing resources. However, the envisioned integration creates many new challenges, among which mobility management is a critical one. Simply applying existing radio access oriented mobility management schemes leads to poor performance due to the highly overlapped coverages of multiple BSs in the proximity of the user, and the co-provisioning of radio access and computing services of the MEC-enabled BSs. In this paper, we develop a novel user-centric energy-aware mobility management (EMM) scheme, in order to optimize the radio access and computation performance under the mobile user's long-term energy consumption constraint. Based on Lyapunov optimization and multi armed bandit theories, EMM works in an online fashion without requiring future system state information, and effectively handles the lack of exact environmental state information. Theoretical analysis explicitly takes BS handover and computation migration cost into consideration and gives a bounded deviation on both delay performance and energy consumption compared to the oracle solution with exact and complete future system information. The proposed algorithm also effectively handles the scenario in which candidate BSs vary during the offloading process of a task due to BS on/off. Simulations show that our proposed algorithms can achieve close-to-optimal delay performance while satisfying the energy consumption constraint.**

***Index Terms*—Mobile edge computing, mobility management, Lyapunov optimization, multi-armed bandit, handover cost.**

I. Introduction

Ultra dense networking (UDN) [2] and mobile edge computing (MEC) (a.k.a. fog computing) [3]–[5] are considered as key building blocks for the next generation 5G mobile network. UDN increases the network capacity through the ultra-dense deployment of small cell base stations (BSs), which is viewed as the key technology addressing the so-called 1000x capacity challenge [6]. MEC provides cloud computing and storage resources at the edge of the mobile network, creating significant benefits such as ultra-low latency, intensive computation capabilities while reducing the network congestion, which are necessary for emerging applications such as Internet of things, video stream analysis, augmented reality and connected cars [7].

It is envisioned that endowing each radio access node with cloud functionalities will be a major form of MEC deployment scenarios, i.e., MEC-enabled UDN [8] [9]. However, current studies on UDN and MEC are mostly separate efforts. Despite the enormous potential benefits brought by the integration of UDN and MEC, significant new challenges are created at the same time. A key bottleneck to the overall system performance is mobility management, which is the fundamental function of tracking mobile devices and associating them with appropriate BSs, thereby enabling mobile services (i.e. radio access and computing) to be delivered. Traditionally, mobility management was designed for providing radio access only. Merging UDN and MEC drastically complicates the problem. Simply applying existing solutions leads to poor mobility management due to the highly overlapped coverage areas of multiple BSs in the vicinity and the co-provisioning of radio access and computing services. In particular, mobility management for MEC in UDN faces the following three main challenges:

1) The first challenge is the lack of accurate information (radio access load, computation load and latency) of candidate BSs on the user side, especially when mobility management is carried out in a user-centric manner. If the user does not know a priori which BS offers the best performance, the mobility management can be very difficult.

2) An even severe challenge is the unavailability of future information (including future tasks, candidate BSs, channel conditions, available edge computing resources, etc.). Since the mobile user has limited battery power, the long-term energy budget couples the short-term mobility management decisions across time, and yet the decisions have to be made without foreseeing the future.

3) Moreover, UDN is a very complex and volatile network environment due to the fact that many small cell BSs are owned, deployed and managed by end-users. In addition, the operator often implements BS sleeping techniques for energy saving purposes. As a result, the set of candidate BSs can be changing over time, thus demanding for a mobility management algorithm that can fast track the optimal BS for maximizing the MEC performance.

A. Related Work

Mobile edge computing has received an increasing amount of attentions recently, see [5] for a comprehensive survey. A central theme of many prior studies is to design offloading policies and resource management schemes, i.e. what/when/how to offload a user's workload from its device to the edge system or cloud, and how much radio and computing resources should be allocated to each user. For a single-user MEC system, an energy-optimal binary offloading policy is

Y. Sun and S. Zhou are with the Department of Electronic Engineering, Tsinghua University, China. Email: sunyx15@mails.tsinghua.edu.cn, sheng.zhou@tsinghua.edu.cn.

J. Xu is with the Department of Electrical and Computer Engineering, University of Miami, USA. Email: jiexu@miami.edu.

Part of this work has been accepted by IEEE ICC 2017 [1].

proposed in [10] by comparing the energy consumption of local execution and offloading, while a delay-optimal task scheduling policy with random task arrivals is proposed in [11] based on Markov Decision Process (MDP). For multi-user MEC systems, both centralized [12] [13] and distributed [14] resource management schemes are studied for joint radio and computational resource management, in order to maximize system-level performance metric. However, most of the existing works considers a single MEC server case, and does not consider the user mobility issue.

Mobility management has been extensively investigated in LTE systems (see [15] and references therein). These solutions work efficiently in less-densified heterogeneous networks, but may bring new problems such as frequent handover and the Ping-Pong effect when the network density becomes high [16]. To address this challenge, an energy-efficient association and power control policy is proposed in [17], while a learning-based mobility management scheme is proposed in [18] based on the multi-armed bandits (MAB) theory [19]. Both schemes work in a user-centric manner, which has been an emerging trend of designing mobility management, particularly for the future 5G standard [20]. However, all these works merely consider the radio access. Endowing BSs with MEC capabilities requires new mobility management solutions.

There are a few works considering service migration in MEC. An optimal computation migration policy is designed in [21] in order to reduce migration cost while maintaining good user quality of service. Based on MDP, a threshold based structure of the optimal policy is shown through simulation. The existence of the optimal threshold policy is further proved in [22]. However, the radio access aspect has not been considered in the existing work.

Motivated by the limitations of the current literature, we design user-centric mobility management algorithms in MEC-enabled UDN in this work. Our work aims to solve a finite horizon average delay minimization problem under a long-term energy budget constraint, with the challenges of lacking both accurate future information and current BS-side information. By integrating Lyapunov optimization technique [23] and MAB theory [19], our proposed algorithms can provide strong performance guarantee. Different from the conference version of this work [1], we introduce a more general model considering transmission delay and BS handover cost, and provide new theoretical analysis and simulation results. Moreover, we also develop a new algorithm based on the volatile MAB (VMAB) framework [24] to handle varying BS sets during task offloading.

B. Contributions

1) We develop a novel energy-aware user-centric mobility management scheme, called EMM, to overcome the aforementioned challenges by leveraging the combined power of Lyapunov optimization and MAB theories. The proposed EMM algorithms can deal with various practical deployment scenarios, including those in which the user has limited BS-side information and the set of serving BSs dynamically changes.

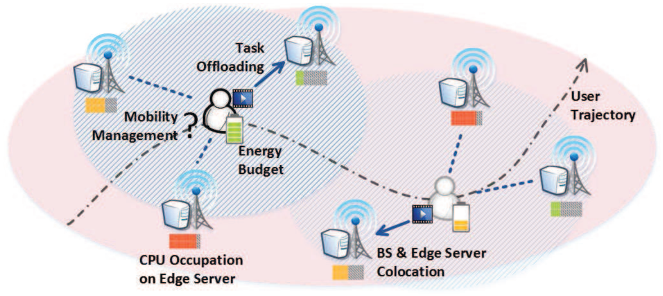


Fig. 1. Mobility management in MEC-enabled UDN.

2) We rigorously characterize the performance of the proposed EMM algorithms. We prove that the EMM algorithms can achieve the optimal performance within a bounded deviation without requiring future system information, while satisfying the long-term energy budget constraint. Moreover, we quantify the performance loss due to learning the BS-side information in terms of the learning regret, explicitly taking into account the additional costs caused by BS handover and computation migration, and varying candidate BSs.

3) Extensive simulations are carried out to evaluate the performance of the proposed algorithms and validate our theoretic findings. The results confirm that our proposed algorithm can achieve close-to-optimal delay performance compared to the oracle solution with exact and complete future system information, while satisfying user's energy consumption constraint, thereby significantly improving the MEC performance in UDN. The simulations also reveal the impact of design parameters on the system performance, thereby providing guidelines for real-world deployment.

The rest of this paper is organized as follows. We describe the system model and formulate the problem in Section II. Section III and IV develop EMM algorithms and conduct performance analysis. Section V extends the algorithm to handle varying BS sets. Simulation results are provided in Section VI, followed by the conclusion in Section VII.

II. System Model and Problem Formulation

A. Network Model

We consider a network with N BSs indexed by $\mathcal{N} = \{1, 2, ..., N\}$, as shown in Fig. 1. Each BS is endowed with cloud computing functionalities, which is considered as one of the main deployment scenarios for MEC [8] [9]. We focus on a representative mobile user moving in the network, who generates totally M computation tasks over time, and these tasks are offloaded to the BS to compute. Let L_m denote the location where task m is generated. No prior knowledge about the user trajectory is required. In other words, our work is applicable to any mobility model, such as the widely used random waypoint model or other mobility models described in [25].

Multiple BSs can provide service to users at any location L_m due to the dense deployment. Denote $\mathcal{A}(L_m) \subseteq \mathcal{N}$ as the set of BSs that cover location L_m. After task m is generated, the mobility management scheme makes decisions on which

BS serves the user, among the set of BSs $\mathcal{A}(L_m)$. In LTE standards, the user is responsible for handover measurement and feedback, while the BS or the Mobility Management Entity (MME) makes the handover decisions [15]. Recently, the network is evolved from the cell-centric architecture into a user-centric one where user-centric mobility management is becoming promising for the 5G standard [20]. Therefore, we design user-centric mobility management schemes, i.e., the user makes the BS association and handover decisions. Moreover, we focus on a local computation scenario meaning that the associated BS is responsible for providing both radio access and edge computing services without further offloading the computation tasks to other BSs or the remote cloud.

B. Computation Task and Service Model

A widely used three-parameter model [5] is adopted to describe each computation task: input data size, computation intensity and completion deadline. Denote $\lambda_m \in [0, \lambda_{\text{max}}]$ as the input data size of task m (in bits) that needs to be offloaded, where λ_{max} is the maximum possible input data size. Let $\gamma_m \in [0, \gamma_{\text{max}}]$ denote the computation intensity of task m (in CPU cycles per bit) with maximum value γ_{max}, which indicates how many CPU cycles are required to compute one bit input data. Moreover, denote D_m as the completion deadline of task m (in seconds). If the task is finished after the deadline, then the computation result is useless to the user. However, the user still prefers to receive the result as soon as possible to gain higher utilities. Each computation task is relatively large and hence can be further divided into many subtasks that must be processed in sequence (e.g. computing the subsequent subtasks requires the results of the previous subtasks). Taking video stream analytics as an example, a task can be object detection from a video stream. The analysis is operated on the edge server using the Hadoop MapReduce framework [26]. Since a video stream may be relatively long, it can be further divided into many shorter video clips, each having a number of video frames. Let $K_m \leq \bar{K}$ be the number of subtasks of task m, where $\bar{K}$ is maximum number of subtasks. Assume that subtasks are of the equal size λ_0 for analytical simplicity (hence $\lambda_m = K_m\lambda_0$). Nevertheless, our framework can be easily extended to handle subtasks of heterogeneous sizes.

Each BS $n \in \mathcal{N}$ is equipped with an MEC server of maximum CPU frequency F_n (in CPU cycles per second), and can provide computation services for multiple tasks from multiple users simultaneously using processor sharing. We use computation capability $f_{m,n}$ to describe the CPU frequency that BS n can allocate to task m. The allocated speed $f_{m,n}$ depends on many factors on the BS side, such as the maximum CPU frequency F_n, the current total workload intensity, etc. We assume that $f_{m,n}$ does not change during one task but can change across tasks. If BS n is selected to compute a subtask of size λ_0 and computation intensity γ_m, then given the allocated CPU frequency $f_{m,n}$, the computation delay is

$$d_c(m,n) = \frac{\lambda_0 \gamma_m}{f_{m,n}}. \tag{1}$$

C. Communication and Energy Consumption Model

The input data is transmitted from the user to the serving BS through the wireless uplink channel. Denote $H_{m,n}$ as the channel gain between the user at location L_m and BS $n \in \mathcal{A}(L_m)$. We assume that during the computation of each task m, the user does not move much and hence $H_{m,n}$ is constant. Nevertheless, if the user moves considerably, we can treat one task as multiple subtasks, and for each subtask the user stays more or less at the same location. Given the transmission power P_{tx} of the user, the maximum achievable uplink transmission rate is given by the Shannon's theorem:

$$r(m,n) = W \log_2 \left(1 + \frac{P_{tx} H_{m,n}}{\sigma^2 + I_{m,n}}\right), \tag{2}$$

where W is the channel bandwidth, σ^2 is the noise power and $I_{m,n}$ is the inter-cell interference of BS n while offloading task m. The transmission delay for sending the input data of size λ_0 to BS n is thus

$$d_t(m,n) = \frac{\lambda_0}{r(m,n)}. \tag{3}$$

Also, the energy consumption for offloading a subtask for task m is therefore

$$e(m,n) = \frac{P_{tx}\lambda_0}{r(m,n)}. \tag{4}$$

Note that in this work we only consider the uplink transmission delay and ignore the downlink transmission delay, which is typically small since the transmission power of the BS is high and the data size of the result is often small. Note that adding the downlink transmission delay into our model only changes the formula of communication delay, but makes no difference on the formulation.

D. Handover and Migration Cost Model

For each computation task m, its subtasks can be offloaded to different BSs, although they still must be computed in sequence. This may be because the user learns that the serving BS's computing capability is weak (we will introduce the learning problem in Section IV) and hence decides to switch to a different BS in its vicinity or BSs can appear or disappear in the transmission range of the user due to BS on/off for energy saving [27]. When consecutive subtasks are processed on different BSs, an additional delay cost is incurred due to the handover procedure and the computation migration. Let C_m be the one-time handover cost for task m. Given the sequences of BSs that serve its subtasks, denoted by $\boldsymbol{a}_m = (a_m^1, a_m^2, ..., a_m^{K_m})$, the overall handover cost for task m is

$$h(m, \boldsymbol{a}_m) = C_m \sum_{k=2}^{K_m} \mathbb{I}\{a_m^k \neq a_m^{k-1}\}, \tag{5}$$

where $a_m^k \in \mathcal{A}(L_m)$ is the serving BS for subtask k of task m, and $\mathbb{I}\{x\}$ is an indicator function with $\mathbb{I}\{x\} = 1$ if event x is true and $\mathbb{I}\{x\} = 0$ otherwise.

E. Problem Formulation

Mobile users often have limited energy budgets (e.g. due to limited battery capacity). Therefore, the objective of the mobile user is to make mobility management decisions (which BS to associate, and when to perform handover) in order to maximize its computation performance (i.e. minimizing computation delay) given its limited energy budget. For task m, the overall delay is

$$D(m, \boldsymbol{a}_m) = \sum_{k=1}^{K_m} d(m, a_m^k) + h(m, \boldsymbol{a}_m), \tag{6}$$

where $d(m, a_m^k) \triangleq d_c(m, a_m^k) + d_t(m, a_m^k)$ is the sum of computation delay and uplink transmission delay for subtask k. The overall energy consumption for processing task m is

$$E(m, \boldsymbol{a}_m) = \sum_{k=1}^{K_m} e(m, a_m^k). \tag{7}$$

Formally, the problem is formulated as follows

$$\textbf{P1:} \quad \min_{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_M} \frac{1}{M}\sum_{m=1}^{M} D(m, \boldsymbol{a}_m) \tag{8}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} E(m, \boldsymbol{a}_m) \le \alpha B \tag{9}$$

$$D(m, \boldsymbol{a}_m) \le D_m, \ \forall m \tag{10}$$

$$a_m^k \in \mathcal{A}(L_m), \forall m, \forall k = 1, 2, ..., K_m. \tag{11}$$

The first constraint (9) states that the total energy consumption is limited by the available battery during the current trip of the user, where $\alpha \in (0, 1]$ indicates the desired capping of energy consumption relative to the total battery capacity B. The second constraint (10) requires that the overall delay for processing task m does not exceed the completion deadline D_m. The last constraint (11) states that the associated BSs are those that cover location L_m.

There are two major challenges to solve problem **P1**. First, optimally solving **P1** requires complete offline information over the entire trip of the user, including parameters of all tasks, user trajectory, traffic intensity of all BSs, etc., which is impossible to acquire in advance. Furthermore, **P1** belongs to integer nonlinear programming. Even if the complete offline information is known a priori, it is still difficult to solve due to the high computation complexity. Therefore, we will propose online algorithms that can efficiently make mobility management decisions without the future information.

F. Oracle Benchmark and Theoretical Upper Bound

In this subsection, we describe an algorithm that knows the complete future information for the next J computation tasks. Albeit impractical, the purpose of introducing this algorithm is merely to provide theoretical upper bounds on the performance of any practical algorithms that do not rely on future information. We will prove later that our proposed algorithm, which does not require future information, achieves close-to-optimal performance compared to this oracle benchmark.

In this algorithm, the entire trip of the user is divided into $R \ge 1$ frames. In each frame, the user generates $J \ge 1$ tasks and hence $M = RJ$. We assume that there is an oracle that provides accurate information of the subsequent J tasks at the beginning of each frame. Given this information, the user can obtain the mobility management decisions for the next J tasks by solving the following J-step lookahead problem,

$$\textbf{P2:} \quad \min_{\boldsymbol{a}_{rJ+1},\ldots,\boldsymbol{a}_{(r+1)J}} \frac{1}{J}\sum_{m=rJ+1}^{(r+1)J} D(m, \boldsymbol{a}_m) \tag{12}$$

$$\text{s.t.} \quad \sum_{m=rJ+1}^{(r+1)J} E(m, \boldsymbol{a}_m) \le \frac{\alpha B}{R} \tag{13}$$

$$\text{constraints (10), (11).} \tag{14}$$

Clearly if $R = 1$, then the J-step lookahead problem is the original offline problem **P1**. Assume that for all $r = 0, 1, ..., R-1$, there exists at least one sequence of mobility management decisions $\boldsymbol{a}_{rJ+1}, ..., \boldsymbol{a}_{(r+1)J}$ that satisfy the constraints of **P2**. Denote g_r^* as the optimal average delay achieved by **P2** in the r-th frame. The minimum long-term average delay achieved by the optimal solution of J-step lookahead problem is thus given by $g^* = \frac{1}{R}\sum_{r=0}^{R-1} g_r^*$, which serves a theoretical upper bound on the performance of any practical mobility management algorithms.

III. Online Mobility Management Framework

In this section, we develop a framework that supports online mobility management requiring only current information. Specifically, when making the mobility management decisions for task m, the user has no information about tasks $m + 1$, $m+2$, ... We will prove that our proposed algorithm achieves close-to-optimal performance compared with the oracle algorithm with J-step lookahead. The information regarding task m can be classified into two categories depending on which entity possesses the information:

- **User-Side State Information**: The user's location L_m, the available candidate BSs $\mathcal{A}(L_m)$, the input data size λ_m and the computation intensity γ_m.
- **BS-Side State Information**: For each BS $n \in \mathcal{A}(L_m)$, the allocated CPU frequency $f_{m,n}$, the uplink channel gain $H_{m,n}$ and the inter-cell interference $I_{m,n}$.

Depending on whether the user has the BS-side state information, we will consider two practical deployment scenarios. In the first scenario, the user knows both user-side state information and BS-side state information exactly. In other words, the user has Global State Information (GSI). In the second scenario, the user only knows the user-side state information. In other words, the user has Local State Information (LSI). In this case, the user needs to learn the BS-side state information in order to make proper mobility management decisions.

Next, we present the online mobility management framework for the scenario with GSI. We will consider LSI in the subsequent sections.

A. EMM-GSI Algorithm

Assume that the serving BS set does not change during one task, then it is clear that if the user has GSI, BS handover and computation migration of subtasks can be avoided. It is straightforward for the user to select the best BS for offloading and computation and stick to the BS for the entire task. Therefore, the sequence of BS association $\boldsymbol{a}_m$ reduces to a single BS association action a_m. We use $D(m,n)$ to denote the overall delay and $E(m,n)$ to denote the overall energy consumption by associating to BS n for task m with GSI.

However, a significant challenge remains of directly solving **P1** since the long-term energy consumption budget couples the mobility management decisions across different tasks: using more energy for the current task will potentially reduce the energy budget available for future uses, and yet the decisions have to be made without foreseeing the future. To address this challenge, we leverage Lyapunov optimization technique which enables us to solve a deterministic problem for each task with low complexity, while adaptively balancing the delay performance and long-term energy consumption.

To guide the mobility management decisions with Lyapunov optimization technique, we first construct a virtual energy deficit queue. Specifically, the dynamics of the energy deficit queue evolves as

$$q(m+1) = \max\{q(m) + E(m,a_m) - \alpha B/M, 0\}, \quad (15)$$

with $q(0) = 0$. The virtual queue length $q(m)$ indicates how far the current energy usage deviates from the battery energy budget. Since the battery capacity of the user device is finite, it is necessary to consider the case with finite tasks and propose an approach that can guarantee the finite worst-case delay performance. Moreover, both the user-side state information and BS-side state information may not follow a well-defined stochastic process. Therefore, we do not make any ergodic assumptions on the state information. Instead, we adopt a non-ergodic version of Lyapunov optimization, which applies to any arbitrary sample path of the task and system dynamics. The algorithm is called EMM-GSI, and is shown in Algorithm 1.

Algorithm 1 EMM-GSI Algorithm

```
1: Input: $L_m$, $\mathcal{A}(L_m)$, $\lambda_m$, $\gamma_m$, and $\forall n \in \mathcal{A}(L_m)$, $f_{m,n}$,
   $H_{m,n}$, $I_{m,n}$ at the beginning of offloading each task $m$.
2: if $m = rJ+1, \forall r = 0,1,...,R-1$ then
3:     $q(m) \leftarrow 0$ and $V \leftarrow V_r$.
4: end if
5: Choose $a_m$ subject to (10), (11) by solving
       (P3) $\min_{n\in\mathcal{A}(L_m)} VD(m,n) + q(m)E(m,n)$.
6: Update $q(m)$ according to (15).
```

Note that EMM-GSI algorithm works in an online fashion, because it requires only the currently available information as the inputs. $V_0, V_1, ..., V_{R-1}$ is a sequence of positive control parameters to dynamically adjust the tradeoff between delay performance and energy consumption over the R frames, each with J periods. Lines 2 - 4 reset the energy deficit virtual queue at the beginning of each frame. Line 5 defines an online optimization problem **P3** to decide the mobility management decisions for each task. The optimization problem aims to minimize a weighted sum of the delay cost and energy consumption where the weight depends on the current energy deficit queue length and is varying over time. A large weight will be placed on the energy consumption if the current energy deficit is large. The energy deficit queue maintains without foreseeing the future guides the mobility management decisions towards meeting the battery energy constraint, thereby enabling online decisions. Note that since there is no BS handover and computation migration, **P3** is equivalent to

$$\min_{n\in\mathcal{A}(L_m)} Vd(m,n) + q(m)e(m,n). \quad (16)$$

Conveniently, we write $z(m,n) \triangleq Vd(m,n) + q(m)e(m,n)$.

B. Performance Bound

In this subsection, we present the performance analysis of EMM-GSI algorithm. Under the feasibility assumption that there exists at least one solution to **P2**, Theorem 1 can be derived as follows.

Theorem 1. *For any fixed integer $J \in \mathbb{Z}_+$ and $R \in \mathbb{Z}_+$ such that $M = RJ$, the following statements hold.*

(1) The average delay performance achieved by EMM-GSI algorithm satisfies:

$$d_G^* \le \frac{1}{R}\sum_{r=0}^{R-1} g_r^* + \frac{UJ}{R}\sum_{r=0}^{R-1}\frac{1}{V_r}, \quad (17)$$

where g_r^* is the optimal average delay of the J-step lookahead problem for frame r, and U is a constant defined as $U \triangleq \frac{1}{2}\max\{(E(m,a_m) - \alpha B/M)^2\}$.

(2) The total energy consumption is within a bounded deviation:

$$e_G^* \le \alpha B + \sum_{r=0}^{R-1}\sqrt{2UJ^2 + 2V_r J g_r^*}. \quad (18)$$

Proof. See Appendix A. □

Theorem 1 shows that using the proposed EMM-GSI algorithm, the worst-case average delay performance is no more than $O(1/V)$ with respect to the optimal average delay achieved by the J-step lookahead problem. Meanwhile, the energy consumption is within a bounded deviation $O(V)$ compared to the given energy budget. Hence, there exists a delay-energy tradeoff of $[O(1/V), O(V)]$. By adjusting V, we can balance the computation delay performance and energy consumption.

IV. Learning with Only LSI

In this section, we consider the scenario that the user has only LSI. We augment our EMM algorithm with an online learning algorithm based on the MAB framework in order to learn the optimal BS (i.e. the solution to **P3**) without requiring the BS-side information initially. Learning the optimal BS incurs additional costs since (1) suboptimal BSs will be selected during the learning process, and (2) BS handover and computation migration is inevitable. We also provide theoretical bounds on the performance loss of the proposed algorithm due to learning.

A. EMM-LSI Algorithm

When the user has only LSI, mobility management is much more difficult since there is no *a priori* information about which BS provides better computation performance while incurring less energy consumption. Specifically, the user cannot directly solve **P3** for each task m since $d(m,n)$ and $e(m,n)$ relies on BS-side information such as $f_{m,n}$, $H_{m,n}$ and $I_{m,n}$, which is now unknown. Therefore, the user has to learn the optimal BS on-the-fly.

A straightforward learning scheme is as follows: the user offloads one subtask of task m to every BS n in $\mathcal{A}(L_m)$ and observes the computation delay $\tilde{d}(m,n)$ and energy consumption $\tilde{e}(m,n)$ (and hence the realized $\tilde{z}(m,n)$). If observations are accurate, namely $\tilde{d}(m,n) = d(m,n)$ and $\tilde{e}(m,n)$ (and hence $\tilde{z}(m,n) = z(m,n)$), then learning can be terminated and the remaining $K_m - |\mathcal{A}(L_m)|$ subtasks of task m will be offloaded to the BS that is the solution to $\min_n \tilde{z}(m,n)$. However, due to the variance in computation intensity, wireless channel state and many other factors, $\tilde{z}(m,n)$ is only a noisy version of $z(m,n)$. In the presence of such measurement variance, this simple learning algorithm can perform very poorly since the user may get trapped in a BS who actually has a large $z(m,n)$. Therefore, a more sophisticated and effective learning algorithm requires continuous learning to smooth out the measurement noise. In fact, mobility management with only LSI manifests a classic sequential decision making problem that involves a critical tradeoff between exploration and exploitation: the user needs to explore the different BSs by offloading subtasks to them in order to learn good estimates of $z(m,n), \forall n \in \mathcal{A}(L_m)$, while at the same time it wants to offload as many subtasks as possible to the a priori unknown optimal BS.

Sequential decision making problems under uncertainties have been extensively studied under the MAB framework and many efficient learning algorithms have been developed that provide strong performance guarantee. In this paper, we augment our EMM algorithm with the widely adopted UCB1 algorithm [19] to learn the optimal BS. Specifically, UCB1 is an index-based algorithm, which assigns an index to each candidate BS and updates the indices of the BSs as more subtasks of a task have been offloaded. Then the next subtask to be offloaded will be offloaded to the BS with the largest index. The index for a BS $n \in \mathcal{A}(L_m)$ is in fact an upper confidence bound on the empirical estimate of $z(m,n)$. Nevertheless, other learning algorithm can also be incorporated in our framework.

The EMM-LSI algorithm is shown in Algorithm 2. The major difference from Algorithm 1 is that instead of solving **P3** exactly, we use the UCB1 algorithm as a subroutine to learn the optimal BS to minimize the objective in **P3**, which is reflected from Lines 5 through 15. Let $\bar{z}_{m,n,k}$ denote empirical sample-mean estimate of $z(m,n)$ after the first k subtasks have been offloaded and their corresponding delay and energy performance have been measured. We use $\theta_{m,n,k}$ to denote the number of subtasks that have been offloaded to BS n up to subtask k. Lines 5-9 is the initialization phase, where one subtask is offloaded to one candidate BS, Lines 10-15 is the continuous learning phase.

Algorithm 2 EMM-LSI Algorithm

```
1:  Input: $L_m$, $\mathcal{A}(L_m)$, $\lambda_m$, $\gamma_m$ at the beginning of offloading
    each task $m$.
2:  if $m = rJ+1, \forall r = 0,1,...,R-1$ then
3:      $q(m) \leftarrow 0$ and $V \leftarrow V_r$.
4:  end if
5:  for $k = 1,...,|\mathcal{A}(L^m)|$ do                    ▷ UCB1 Learning
6:      Connect to each BS $n \in \mathcal{A}(L_m)$ once.
7:      Update $\bar{z}_{m,n,k} = V\tilde{d}(m,n) + q(m)\tilde{e}(m,n)$.
8:      Update $\theta_{m,n,k} = 1$.
9:  end for
10: for $k = |\mathcal{A}(L^m)|+1,...,K_m$ do
11:     Connect to $a_m^k = \arg\min_n\{\bar{z}_{m,n,k} - \beta\sqrt{\frac{2\ln k}{\theta_{m,n,k}}}\}$.
12:     Observe $\tilde{d}(m,a_m^k)$ and $\tilde{e}(m,a_m^k)$.
13:     $\bar{z}_{m,a_m^k,k} \leftarrow \frac{\theta_{m,a_m^k,k}\bar{z}_{m,a_m^k,k} + V\tilde{d}(m,a_m^k) + q(m)\tilde{e}(m,a_m^k)}{\theta_{m,a_m^k,k}+1}$.
14:     $\theta_{m,a_m^k,k} \leftarrow \theta_{m,a_m^k,k} + 1$.
15: end for
16: Update $q(m)$ according to (15).
```

B. Algorithm Performance

In this subsection, we analyze the performance of EMM-LSI. We first bound the gap between the exact solution of **P3** with GSI and the UCB1 learning algorithm with LSI for each task. Understanding the performance loss due to learning for each task is important for understanding the performance of EMM-LSI algorithm for the long-term problem. We adopt the concept of *learning regret* to measure the performance loss due to learning, which is commonly used in the MAB framework. Formally, the learning regret is defined as follows

$$R_m = \mathbb{E}[Z(m,\boldsymbol{a}_m) - Z(m,a_m^*)], \tag{19}$$

where $Z(m,\boldsymbol{a}_m) = VD(m,\boldsymbol{a}_m) + q(m)E(m,\boldsymbol{a}_m)$ is the weighted cost achieved by the sequence of mobility management decisions $\boldsymbol{a}_m$ resulted from UCB1, and $Z(m,a_m^*) = VD(m,a_m^*) + q(m)E(m,a_m^*)$ is the weighted cost achieved by always connecting to the optimal BS a_m^* that solves **P3**.

Although the learning regret of the UCB1 algorithm has been well understood, characterizing the learning regret of UCB1 applied in our setting faces new challenges: the learning regret is a result of not only offloading subtasks to suboptimal BSs but also BS handover and computation migration. Specifically, the learning regret can be decomposed into two terms [28], namely the *sampling regret* and the *handover regret*:

$$R_m = \underbrace{\mathbb{E}[\sum_{k=1}^{K_m} z(m,a_m^k) - Z(m,a_m^*)]}_{\text{sampling regret}} + V\underbrace{\mathbb{E}[h(m,\boldsymbol{a}_m)]}_{\text{handover regret}}. \tag{20}$$

We provide an upper bound on the learning regret of UCB1 considering the handover regret in the following proposition.

Proposition 1. *For task m comprising K_m subtasks, the learning regret R_m is upper bounded as follows:*

$$\begin{aligned} R_m(K_m) \leq & \beta \left[8 \sum_{n \neq a_m^*} \frac{\ln K_m}{\delta_{m,n}} + \left(1 + \frac{\pi^2}{3}\right) \sum_{n \neq a_m^*} \delta_{m,n} \right] \\ & + VC_m (2 \sum_{n \neq a_m^*} \left[\frac{8 \ln K_m}{\delta_{m,n}^2} + 1 + \frac{\pi^2}{3} \right] + 1), \end{aligned} \tag{21}$$

where $\beta = \sup_n \tilde{z}(m,n)$ and $\delta_{m,n} = (Z(m,n) - Z(a_m^*))/\beta K_m$.

Proof. See Appendix B. □

Remark 1. *β is used to normalize the utility function. In real implementation, it is difficult to obtain the exact value of β due to lack of the BS-side state information. However, a reasonably good estimate of β can be obtained based on history data, e.g., setting β as the maximum $\tilde{z}(m,n)$ that has been observed.*

The bound on the learning regret established in Proposition 1 is logarithmic in the number of subtasks K_m. It also implies that **P3** can be approximately solved by UCB1 within a bounded deviation, denoted by W, since K_m is upper bounded by $\bar{K}$. The performance of EMM-LSI can then be derived in Theorem 2.

Theorem 2. *For any fixed integer $J \in \mathbb{Z}_+$ and $R \in \mathbb{Z}_+$ such that $M = RJ$, the following statements hold.*

(1) The average delay performance achieved by EMM-LSI algorithm satisfies:

$$d_L^* \leq \frac{1}{R} \sum_{r=0}^{R-1} g_r^* + \frac{UJ + W}{R} \sum_{r=0}^{R-1} \frac{1}{V_r}. \tag{22}$$

(2) The total energy consumption is within a bounded deviation:

$$e_L^* \leq \alpha B + \sum_{r=0}^{R-1} \sqrt{2[UJ^2 + V_r J g_r^* + WJ]}. \tag{23}$$

Proof. See Appendix C. □

Theorem 2 shows that the proposed EMM-LSI algorithm can provide a strong performance guarantee: even if the user cannot acquire the exact BS-side state information, the average delay performance can still be guaranteed through the proposed algorithm, while the energy consumption is within a bounded deviation of the given energy budget.

C. Implementation Considerations

In the proposed EMM-LSI algorithm, the user keeps learning the optimal BS while offloading all K_m subtasks of task m. Although Proposition 1 provides an upper bound on the performance loss due to continuous learning, in practice, the loss can be large when the one-time handover cost is relatively large. For instance, when the second-best BS has a similar value of $z(m,n)$ as the optimal BS, the UCB1 algorithm can be alternating between these two BSs for many subtasks, thereby incurring a significant handover and migration cost. To circumvent this issue, there are two possible heuristic schemes.

1) The first scheme stops learning after a pre-determined finite number K_s of times of subtask offloading. That is, UCB1 is applied only for the first K_s subtasks. The remaining $K_m - K_s$ subtasks, if any, will all be offloaded to the BS with the lowest value of $\bar{z}_{m,n,K_s}$. Clearly, there is a tradeoff for deciding K_s: if K_s is too small, the probability that a suboptimal BS is found to be optimal is high, and hence, leading to a large cost for offloading the remaining subtasks to the suboptimal BS. On the other hand, if K_s is too large, a large handover cost may be incurred. We will show this tradeoff in our simulation results.

2) The second scheme stops learning when it is determined that the best and second-best BSs have very similar performance. Specifically, the stopping criteria is

$$\bar{z}_{m,n^*,k} - \bar{z}_{m,n^\dagger,k} \leq \epsilon \tag{24}$$

$$\theta_{m,n^*,k} \geq K_0, \theta_{m,n^\dagger,k} \geq K_0, \tag{25}$$

where n^* represents the learned best BS and $n^\dagger$ is the learned second-best BS so far, and ϵ, K_0 are pre-determined parameters.

V. Varying BS Set

In this section, we consider a more general setting in which the set of candidate BSs during the offloading of one task can vary. For example, BSs are turned on/off according to the BS sleeping strategy for energy saving purposes [27] or small cell owner-governed processes. We develop a modified version of the EMM-LSI algorithm, called EMM-LSI-V, based on the VMAB framework and characterize its performance.

A. EMM-LSI-V Algorithm

The varying set of BSs creates a big challenge in learning the optimal BS that solves **P3**. With the conventional UCB1 algorithm, the user has to restart the learning process whenever a new BS appears. Apparently, this learning strategy is very inefficient since it simply restarts the learning process without reusing what has been learned. Although the available BS set changes, the states of other BSs are likely to remain unchanged. Therefore, proper learning algorithms that effectively reuse the already learned information are needed.

To efficiently learn the optimal BS among a varying BS set, we adopt the VMAB framework [24], in which BSs can appear or disappear unexpectedly with unknown lifespan. Define an epoch as the interval in which the available BS set is invariant, and let B_m be the total number of epochs for task m, which is unknown in advance. Note that $B_m = 1, \forall m$ corresponds to the case that we considered in Section IV and hence, the considered scenario in this section is a generalization. The available BS set for epoch $b = 1, 2, ..., B_m$ is denoted as $\mathcal{A}_{m,b}$ and let $\mathcal{A}_m$ be the union of $\mathcal{A}_{m,b}, \forall b = 1, ..., B_m$. To simplify the problem, we assume that each BS only appears once during each task. If a BS appears for the second time, it can be treated as a new BS and re-learned. For each BS $n \in \mathcal{A}_m$, the lifespan is denoted as $[u_n, v_n]$ with $1 \leq u_n, v_n \leq K_m$, which indicates that BS n is present from subtask u_n through subtask v_n. We also denote $K_{m,b}$ as the total number of subtasks of task m completed by the end of epoch b. Clearly, $K_{m,B_m} = K_m$.

The EMM-LSI-V algorithm developed based on volatile UCB1 (VUCB1) learning is proposed in Algorithm 3. In VUCB1 learning, a UCB1-like algorithm is implemented for each epoch. The differences are two-fold. First, the initialization for each epoch (Lines 6-10) only applies to the newly appeared BSs, while the information for the remaining BSs is retained and hence reused. Second, the index term in Line 12 used to guide the subtask offloading decision takes into account the appearance time of the BS.

Algorithm 3 EMM- LSI-V Algorithm

```
1: Input: $\lambda_m$, $\gamma_m$ at the beginning of each $m$.
2: if $t = rJ+1, \forall r = 0, 1, ..., R-1$ then
3:     $q(m) \leftarrow 0$ and $V \leftarrow V_r$
4: end if
5: for $k = 1, ..., K_m$ do                    ▷ VUCB1 Learning
6:     if $k$ is the first block of an epoch then
7:         Input: $\mathcal{A}_{m,b}$
8:         Connect to each first appeared BS $n \in \mathcal{A}_{m,b}$ once
9:         Update $\bar{z}_{m,n,k} = V\tilde{d}(m,n) + q(m)\tilde{e}(m,n)$
10:        Update $\theta_{m,n,k} = 1$
11:    else
12:        Choose $a_m^k = \arg\min_n\{\bar{z}_{m,n,k} - \beta\sqrt{\frac{2\ln(k-u_n)}{\theta_{m,n,k}}}\}$
13:        Observe $\tilde{d}(m,a_m^k)$ and $\tilde{e}(m,a_m^k)$
14:        $\bar{z}_{m,a_m^k,k} \leftarrow \frac{\theta_{m,a_m^k,k}\bar{z}_{m,a_m^k,k} + V\tilde{d}(m,a_m^k) + q(m)\tilde{e}(m,a_m^k)}{\theta_{m,a_m^k,k}+1}$
15:        $\theta_{m,a_m^k,k} \leftarrow \theta_{m,a_m^k,k} + 1$
16:    end if
17: end for
18: Update $q(m)$ according to (15).
```

B. Algorithm Performance

We characterize the performance of the VUCB1 learning as follows. Let $a^*_{m,b}$ as the optimal BS at epoch b for task m. The learning regret is thus

$$R_m = \underbrace{\sum_{b=1}^{B_m} \mathbb{E}[\sum_{k=K_{m,b-1}+1}^{K_{m,b}} z(m,a_m^k) - Z(m,a^*_{m,b})]}_{\text{sampling regret}} + V\underbrace{\mathbb{E}[h(m,\boldsymbol{a}_m)]}_{\text{handover regret}}. \quad (26)$$

Proposition 2. *For task m comprising K_m subtasks, if there are B_m epochs, the total regret R_m of VUCB1 is $O(B\ln K)$.*

Proof. See Appendix D. □

Proposition 2 states that VUCB1 learning can provide a bounded deviation, defined as W', from exactly solving **P3**. Therefore, our EMM-LSI-V algorithm can still provide strong performance guarantee by substituting W with W' in Theorem 2.

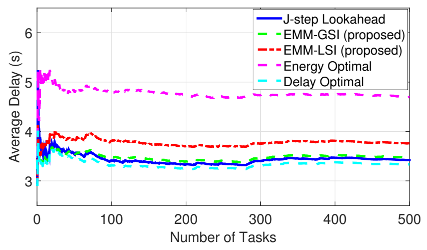
VI. Simulations

In this section, we evaluate the performance of the proposed EMM algorithms and verify the theoretical results through simulations. We simulate a 2km×2km square area with 36 BSs deployed on a regular grid network. The user trajectory is generated by the random walk model. Since BSs are densely deployed, there are multiple available BSs to provide service for a user at any location, and the user will choose one BS for task/subtask offloading. The association radius is set to be 400m. The wireless channel gain is modeled as $H_{m,n} = 25.3 + 37.6 \times \log d$, as suggested in [29]. Besides, channel bandwidth W = 20M/Hz, noise power $\sigma^2 = 2 \times 10^{-13}$W, and transmit power $P_{tx} = 0.5$W.

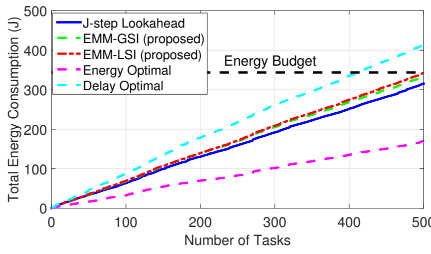
We consider a video stream analysis application with totally $M = 500$ video tasks generated during the entire trip. Each subtask is a one-second video clip. According to [26], we set $\lambda_0 = 0.62$Mbits, which is the data size of a one-second QCIF format video with 176×144 video resolution, 24.8k pixels per frame and 25 fps (frame per second). Each video is set to be 1min to 2min long, i.e., K_m is uniformly selected from $\{60, 61, ..., 120\}$, and thus the input data size satisfies $\lambda_m \in [37.2, 74.4]$ Mbits. Each subtask has completion deadline 150ms, and the computation intensity is uniformly distributed with $\gamma_m \in [500, 1000]$ cycles/bit. Each MEC sever is equipped with multiple CPU cores, and the sum frequency F_n = 25GHz. The available computation capability for each task follows uniform distribution with $f_{m,n} \in [0, F_n]$ GHz. In addition, one-time handover cost C_m = 5ms, and battery capacity B = 1200J.

We introduce three benchmark algorithms to evaluate the performance of the proposed EMM algorithms: 1) **Delay Optimal**: the user always associates with the BS with the lowest delay and disregards the energy consumption constraint. 2) **Energy Optimal**: the user always associates with the BS with the best channel condition without considering the delay performance. 3) ***J*-step Lookahead**: this is the oracle benchmark algorithm described in Section II-F. We set $J = 5$ and thus $R = 100$. Note that solving the J-step lookahead problem is extremely computationally complex even if the future information is known.

Fig. 2 compares the average delay performance and total energy consumption over M tasks of EMM-GSI, EMM-LSI and three benchmark algorithms. Here we set 30% observation variance in LSI scenario and let EMM-LSI algorithm stop learning after offloading $K_s = 20$ subtasks to avoid frequent BS handover and computation migration, as discussed in Section IV-C. As can be seen, Delay Optimal algorithm achieves the best delay performance at the cost of violating the energy budget constraint. Energy Optimal algorithm uses energy conservatively by always connecting to the BSs with the best channel condition. However, the resulting average delay is significantly higher than other algorithms. Both J-step Lookahead and our two EMM algorithms satisfy the energy consumption constraint while keeping the delay low. Among them, J-step Lookahead is the best as expected, with the help of future information. EMM-GSI algorithm achieves close delay performance to J-step Lookahead, while the delay



(a) Average delay cost



(b) Total energy consumption

Fig. 2. Performance of EMM ($V = 0.003$, $\alpha B = 350$J, $K_s = 20$, 30% observation variance)

performance of EMM-LSI algorithm is just slightly worse than EMM-GSI algorithm.

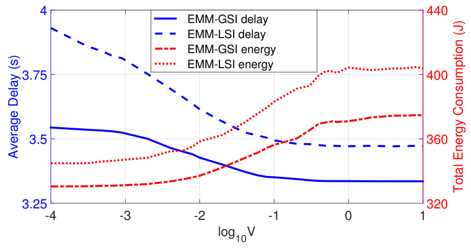
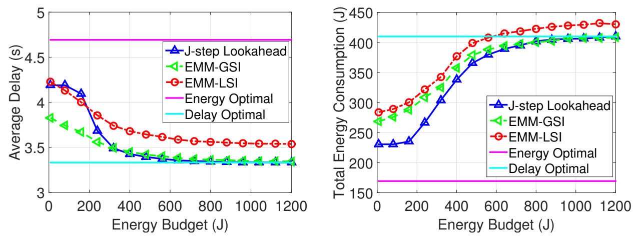


Fig. 3. Impact of V ($\alpha B = 350$J, $K_s = 20$, 30% observation variance)

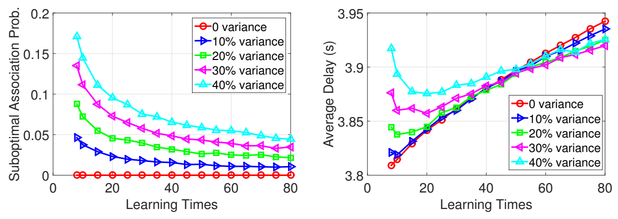
Fig. 3 shows the impact of control parameter V on the average delay performance and total energy consumption. By increasing V from 10^{-4} to 10, both EMM-GSI and EMM-LSI algorithms care more about delay performance, thus the average delay decreases. However, with less concern on energy, the total energy consumption increases and will finally exceed the given budget. The delay-energy performance follows $[O(1/V), O(V)]$ tradeoff, which verifies Theorem 1 and Theorem 2. Meanwhile, the results also provide guidelines for selecting V in real deployment: under the energy budget constraint, we should choose a V that can minimize the average delay performance.

By varying the energy capping parameter α from 1% to 100%, we explore the impact of energy budget on the average delay performance and total energy consumption, as shown in Fig. 4. When the energy budget is large, EMM-GSI achieves the optimal delay performance since the energy constraint is always satisfied, while EMM-LSI incurs additional performance loss due to the learning process. When the energy budget is too low, it is possible that there is no feasible solution and thus the energy constraint is violated. In between, both EMM algorithms make tradeoff between delay performance and energy consumption, and the performance of EMM-GSI is very close to the J-step Lookahead benchmark.



(a) Average delay cost (b) Total energy consumption

Fig. 4. Impact of energy budget αB ($V = 0.003$, $K_s = 20$, 30% observation variance)



(a) Probability of connecting suboptimal BS after learning (b) Average delay performance

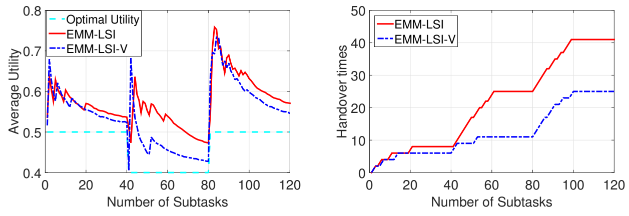
Fig. 5. Impact of learning times K ($V = 0.003$, $\alpha B = 350$J)

We further evaluate the impact of the number of subtasks K_s used for learning in EMM-LSI for implementation considerations. Since the maximum number of available BSs in the whole duration of M tasks is 7 and the learning algorithm needs to associate with each BS for at least one time, we set learning time K_s vary from 8 to 80. We carry out the simulation under different observation variance, and repeat 10 times for average. Fig. 5(a) shows the probability of connecting to a suboptimal BS after using K_s subtasks to learn. When there is no observation variance, the user can always select the optimal BS after connecting to each available BS once. When the observation variance increases, the probability of connecting to a suboptimal BS increases. However, as K_s increase, the probability of connecting to a suboptimal BS decreases drastically. Fig. 5(b) shows the impact of K_s on average delay performance. With K_s increasing, the average delay decreases first and then increases, except the case with zero variance where learning always increases both sampling regret and handover regret. This is because when K_s is small. the probability of connecting to a suboptimal BS after learning is large, which leads to high additional cost. When K_s is large, the frequent handover increases the handover regret and thus degrades the delay performance. Therefore, learning time K_s should be carefully selected to tradeoff the aforementioned two factors. For example, under 30% observation variance, $K_s = 20$ can obtain the best delay performance.

Finally, we compare the proposed EMM-LSI-V algorithm with EMM-LSI under dynamic BS set. We illustrate the results by dividing one task into 3 epochs. The available BSs and their normalized utility (defined in **P3**) is shown in Table I. In epoch 2, there appears an optimal BS and a suboptimal BS, while in epoch 3, an optimal BS disappears and a suboptimal

TABLE I
AVAILABLE BSS AND NORMALIZED UTILITY

Index of BS	1	2	3	4	5
Normalized utility	0.5	0.8	0.4	0.9	0.7
Epoch 1	√	√	–	–	–
Epoch 2	√	√	√	√	–
Epoch 3	√	√	×	√	√



(a) Average utility (b) Handover times

Fig. 6. EMM-LSI-V algorithm vs. EMM-LSI algorithm

BS appears. Each epoch has 40 subtasks and the learning time K_s of both algorithms is set to be 20. Fig. 6 shows that EMM-LSI-V algorithm converges faster than EMM-LSI algorithm, while efficiently reducing the handover times. This is because EMM-LSI-V algorithm retains the information of remaining BSs while EMM-LSI algorithm restarts the learning process whenever there is a change.

VII. CONCLUSIONS

In this paper, we studied the mobility management problem for MEC-enabled UDN. We developed a novel user-centric mobility management framework and design mobility management algorithms, called EMM, that can be applied to both GSI and LSI scenarios by integrating Lyapunov optimization and MAB techniques. Taking BS handover and computation migration cost into consideration, we proved that our proposed algorithms can optimize the delay performance while approximately satisfying the user's energy consumption budget. Furthermore, we proposed a generalized EMM algorithm that can handle varying BS sets based on the VMAB framework, and analyzed its performance. Future research direction includes designing mobility management schemes for high mobility scenarios.

REFERENCES

[1] J. Xu, Y. Sun, L. Chen, and S. Zhou, "E2M2: Energy efficient mobility management in dense small cells with mobile edge computing," *arXiv preprint arXiv:1701.07363*, 2017.

[2] T. Q. Quek, G. de la Roche, I. Guvenc, and M. Kountouris, *Small cell networks: Deployment, PHY techniques, and resource management*. Cambridge University Press, 2013.

[3] ETSI, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing industry initiative*. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *IEEE Commun. Surveys Tuts.*, submitted for publication.

[6] S. Chen and J. Zhao, "The requirements, challenges, and technologies for 5g of terrestrial mobile telecommunication," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 36–43, May. 2014.

[7] ETSI. Mobile edge computing: Service scenatios. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC-IEG/001_099/004/01.01.01_60/gs_MEC-IEG004v010101p.pdf

[8] ——. Mobile edge computing: Technical requirements. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf

[9] TROPIC. Tropic main achievements and exploitation prospects. [Online]. Available: http://www.ict-tropic.eu/documents/others/TROPIC_Template_RAS-2.pdf

[10] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sept. 2013.

[11] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[12] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2016.

[13] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, submitted for publication.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[15] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, "Mobility management for femtocells in lte-advanced: key aspects and survey of handover decision algorithms," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 64–91, 1st Quarter 2014.

[16] D. Lopez-Perez, I. Guvenc, and X. Chu, "Mobility management challenges in 3gpp heterogeneous networks," *IEEE Commun. Mag.*, vol. 50, no. 12, Dec. 2012.

[17] J. Park, S. Y. Jung, S. L. Kim, M. Bennis, and M. Debbah, "User-centric mobility management in ultra-dense cellular networks under spatio-temporal dynamics," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, Dec. 2016, pp. 1–6.

[18] C. Shen, C. Tekin, and M. van der Schaar, "A non-stochastic learning approach to energy efficient mobility management," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3854–3868, Dec. 2016.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[20] S. Chen, F. Qin, B. Hu, X. Li, and Z. Chen, "User-centric ultra-dense networks for 5g: challenges, methodologies, and directions," *IEEE Wireless Commun.*, vol. 23, no. 2, pp. 78–85, Apr. 2016.

[21] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, to appear.

[22] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Baltimore, MD, Oct. 2014, pp. 835–840.

[23] M. J. Neely, *Stochastic network optimization with application to communication and queueing systems*. San Rafael, CA, USA: Morgan & Claypool Publishers, 2010.

[24] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Social network search as a volatile multi-armed bandit problem," *HUMAN*, vol. 2, no. 2, p. 84, 2013.

[25] S. Batabyal and P. Bhaumik, "Mobility models, traces and impact of mobility on opportunistic routing algorithms: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1679–1707, 3rd Quater 2015.

[26] A. Anjum, T. Abdullah, M. Tariq, Y. Baltaci, and N. Antonopoulos, "Video stream analysis in clouds: An object detection and classification framework for high performance video analytics," *IEEE Trans. Cloud Comput.*, to appear.

[27] S. Zhang, J. Gong, S. Zhou, and Z. Niu, "How many small cells can be turned off via vertical offloading under a separation architecture?" *IEEE Trans. Wireless Commun.*, vol. 14, no. 10, pp. 5440–5453, Oct. 2015.

[28] R. Agrawal, M. Hedge, and D. Teneketzis, "Asymptotically efficient adaptive allocation rules for the multiarmed bandit problem with switching cost," *IEEE Trans. Autom. Control*, vol. 33, no. 10, pp. 899–906, 1988.

[29] 3GPP, "Further advancements for e-utra physical layer aspects," TR 36.814.

Appendix A Proof of Theorem 1

For notational convenience, we define $y(m) = E(m, a_m) - \alpha B/M$. According to the energy deficit queue in (15), it is easy to see

$$q(m+1) - q(m) \geq y(m). \tag{27}$$

Summing the above over $m = rJ+1, ..., (r+1)J$, using the law of telescoping sums, we get

$$\sum_{t=rJ+1}^{(r+1)J} y(m) \leq q((r+1)J+1) - q(rJ+1), \tag{28}$$

where $q(rJ+1) = 0$ and $q((r+1)J+1)$ is the queue length before reset in frame $r+1$. In what follows, we try to bound $q((r+1)J+1)$.

Define the Lyapunov function $L(q(m))$ as

$$L(q(m)) \triangleq \frac{1}{2} q^2(m). \tag{29}$$

Moreover, we define the 1-slot Lyapunov drift $\Delta_1(m)$ as:

$$\Delta_1(m) = L(q(m+1)) - L(q(m)), \tag{30}$$

where a "slot" refers to the duration of offloading and computation for a task.

Therefore, the 1-slot drift-plus-penalty function can be expressed as $\Delta_1(m) + VD(m, a_m)$, where $V > 0$ is a control parameter that affects the tradeoff between delay performance and energy consumption.

According to the definition of energy deficit queue in (15), squaring the queuing dynamics equation results in the following bound

$$\begin{aligned} q^2(m+1) &\leq (q(m) + y(m))^2 \\ &= q^2(m) + y^2(m) + 2q(m)y(m). \end{aligned} \tag{31}$$

Therefore, the 1-slot Lyapunov drift $\Delta_1(m)$ satisfies

$$\Delta_1(m) = L(q(m+1)) - L(q(m)) \leq \frac{1}{2} y^2(m) + q(m)y(m). \tag{32}$$

Now define U as a positive constant that upper bounds $\frac{1}{2}y^2(m)$. Such a constant exists under the assumption that $y(m)$ is deterministically bounded. By adding $VD(m, a_m)$ at both sides of (32), we can obtain

$$\begin{aligned} &\Delta_1(m) + VD(m, a_m) \\ &\leq U + VD(m, a_m) + q(m)y(m). \end{aligned} \tag{33}$$

Define the J-slot Lyapunov drift as $\Delta_J(rJ) \triangleq L(q((r+1)J+1)) - L(q(rJ+1))$, we have

$$\begin{aligned} &\Delta_J(rJ) + V \sum_{m=rJ+1}^{(r+1)J} D(m, a_m) \\ \leq& UJ + V \sum_{m=rJ+1}^{(r+1)J} D(m, a_m) + \sum_{m=rJ+1}^{(r+1)J} q(m)y(m) \\ =& UJ + V \sum_{m=rJ+1}^{(r+1)J} D(m, a_m) + \sum_{m=rJ+1}^{(r+1)J} q(rJ+1)y(m) \\ &+ \sum_{m=rJ+1}^{(r+1)J} (q(m) - q(rJ+1))y(m). \end{aligned} \tag{34}$$

Let $y_{max} \geq 0$ denote the maximum positive value of $y(m)$ for all m (otherwise $y_{max} = 0$), i.e., $q(m+1) - q(m) \leq y_{max}$. Thus, for $m = rJ+1, ..., (r+1)J$,

$$q(m) - q(rJ+1) \leq (m - (rJ+1))y_{max}. \tag{35}$$

The last term on the right hand side of (35) satisfies

$$\begin{aligned} &\sum_{m=rJ+1}^{(r+1)J} (q(m) - q(rJ+1))y(m) \\ \leq& \sum_{m=rJ+1}^{(r+1)J} (m - (rJ+1))y_{max}^2 \\ =& \frac{J(J-1)}{2} y_{max}^2 \leq J(J-1)U. \end{aligned} \tag{36}$$

The right hand side of (35) is bounded by

$$\begin{aligned} &\Delta_J(rJ) + V \sum_{m=rJ+1}^{(r+1)J} D(m, a_m) \\ \leq& UJ^2 + V \sum_{m=rJ+1}^{(r+1)J} D(m, a_m). \end{aligned} \tag{37}$$

By applying EMM-GSI algorithm on the left-hand side and considering the optimal J-step lookahead algorithm on the right-hand side, we obtain the following

$$\Delta_J(rJ) + V_r \sum_{m=rJ+1}^{(r+1)J} d_G^*(m) \leq UJ^2 + V_r J g_r^*, \tag{38}$$

where $d_G^*(m)$ is the delay achieved by EMM-GSI algorithm for task m.

Therefore,

$$q((r+1)J+1) = \sqrt{2\Delta_J(rJ)} \leq \sqrt{2(UJ^2 + V_r J g_r^*)}. \tag{39}$$

Substituting (39) into (28), we have

$$\sum_{m=rJ+1}^{(r+1)J} y(m) \leq \sqrt{2(UJ^2 + V_r J g_r^*)}. \tag{40}$$

Therefore,

$$\sum_{m=rJ+1}^{(r+1)J} e_G^*(m) \leq \alpha B/R + \sqrt{2(UJ^2 + V_r J g_r^*)}, \tag{41}$$

where $e_G^*(m)$ is the energy consumption achieved by EMM-GSI algorithm for task m. By summing over $r = 0, 1, ..., R-1$ we prove part (2) of Theorem 1.

By dividing both sides of (38) by V_r, it follows that

$$\sum_{m=rJ+1}^{(r+1)J} d_G^*(m) \leq Jg_r^* + \frac{UJ^2}{V_r}. \tag{42}$$

Thus, by summing over $r = 0, 1, ..., R-1$ and dividing both sides by RJ, we prove part (1) of Theorem 1.

Appendix B
Proof of Proposition 1

The proof follows the similar idea of [19] and the main difference is that we also bound the handover regret.

Since we only focus on the regret in one task, we omit m for notation convenience. The sampling regret SR can be written as

$$\begin{aligned} \text{SR} &= \mathbb{E}[\sum_{k=1}^{K} z(a^k) - Z(a^*)] \\ &= \mathbb{E}[\sum_{n\in\mathcal{A}(L^m)} \theta_{n,K}\frac{Z(n)}{K} - \theta_{a^*,K}\frac{Z(a^*)}{K}] \\ &= \sum_{n\neq a^*} \beta\delta_n \mathbb{E}[\theta_{n,K}]. \end{aligned} \tag{43}$$

We first bound $\theta_{n,K}$. Let $c_{k,s} = \sqrt{2\ln k/s}$, l be any positive integer, and $z' = z/\beta$ is the normalized utility. We have

$$\begin{aligned} \theta_{n,K} &= 1 + \sum_{k=A+1}^{K} \mathbb{I}\{a^k = n\} \\ &\leq l + \sum_{k=A+1}^{K} \mathbb{I}\{a^k = n, \theta_{n,k-1} \geq l\} \\ &\leq l + \sum_{k=A+1}^{K} \mathbb{I}\{\max_{0<s<k} \bar{z}'_{a^*,s} - c_{k,s} \geq \min_{l\leq s_n<k} \bar{z}'_{n,s_n} - c_{k,s_n}\} \\ &\leq l + \sum_{k=1}^{\infty}\sum_{s=1}^{k-1}\sum_{s_n=l}^{k-1} \mathbb{I}\{\bar{z}'_{a^*,s} - c_{k,s} \geq \bar{z}'_{n,s_n} - c_{k,s_n}\}. \end{aligned} \tag{44}$$

$\mathbb{I}\{\bar{z}'_{a^*,s} - c_{k,s} \geq \bar{z}'_{n,s_n} - c_{k,s_n}\}$ implies that at least one of the following three equations hold

$$\bar{z}'_{a^*,s} \geq Z(a^*)/\beta K + c_{k,s}, \tag{45}$$

$$\bar{z}'_{n,s_n} \leq Z(n)/\beta K - c_{k,s_n}, \tag{46}$$

$$Z(a^*)/\beta K > Z(n)/\beta K - 2c_{k,s_n}. \tag{47}$$

By using Chernoff-Hoeffding bound, we have

$$\mathbb{P}\{\bar{z}'_{a^*,s} \geq Z(a^*)/\beta K + c_{k,s}\} \leq e^{-4\ln k} = k^{-4}, \tag{48}$$

$$\mathbb{P}\{\bar{z}'_{n,s_n} \leq Z(n)/\beta K - c_{k,s_n}\} \leq k^{-4}. \tag{49}$$

When $l \geq \lceil \frac{8\ln K}{\delta_n^2} \rceil$, (47) not holds because

$$\begin{aligned} &Z(a^*)\beta K - Z(n)\beta K + 2c_{k,s_n} \\ =& Z(a^*)\beta K - Z(n)\beta K + 2\sqrt{2\ln k/s_n} \\ \leq& Z(a^*)\beta K - Z(n)\beta K + \delta_n = 0. \end{aligned} \tag{50}$$

Then for any $n \neq a^*$, we have

$$\begin{aligned} \mathbb{E}[\theta_{n,K}] \leq& \lceil \frac{8\ln K}{\delta_n^2} \rceil + \sum_{k=1}^{\infty}\sum_{s=1}^{k-1}\sum_{s_n=l}^{k-1} (\mathbb{P}\{\bar{z}'_{a^*,s} \geq Z(a^*)/\beta K \\ &+ c_{k,s}\} + \mathbb{P}\{\bar{z}'_{n,s_n} \leq Z(n)/\beta K - c_{k,s_n}\}) \\ \leq& \lceil \frac{8\ln K}{\delta_n^2} \rceil + \sum_{k=1}^{\infty}\sum_{s=1}^{k-1}\sum_{s_n=l}^{k-1} 2k^{-4} \qquad (51) \\ \leq& \frac{8\ln K}{\delta_n^2} + 1 + \frac{\pi^2}{3}. \qquad (52) \end{aligned}$$

The upper bound of sampling regret is

$$\begin{aligned} \text{SR} &= \sum_{n\neq a^*} \beta\delta(n)\mathbb{E}[\theta_{n,K}] \\ &\leq \beta\left[8\sum_{n\neq a^*}\frac{\ln K}{\delta_n} + \left(1 + \frac{\pi^2}{3}\right)\sum_{n\neq a^*}\delta_n\right]. \end{aligned} \tag{53}$$

The upper bound of handover regret is

$$\begin{aligned} \text{HR} &= V\mathbb{E}[h(m, \boldsymbol{a}_m)] \\ &= VC\mathbb{E}[\sum_{k=2}^{K} \mathbb{I}\{a^k \neq a^{k-1}\}] \\ &= VC\sum_{n\in\mathcal{A}(L^m)} \mathbb{E}[\sum_{k=2}^{K}\mathbb{I}\{a^k = n, a^{k-1} \neq n\}]. \end{aligned} \tag{54}$$

Let $S_n = \sum_{k=2}^{K}\mathbb{I}\{a^k = n, a^{k-1} \neq n\}$ count the handover times from BS n to other BSs. Then

$$\begin{aligned} \text{HR} &= VC(\sum_{n\neq a^*}\mathbb{E}[S_n] + \mathbb{E}[S_{a^*}]) \\ &\leq VC(2\sum_{n\neq a^*}\mathbb{E}[S_n] + 1) \leq VC(2\sum_{n\neq a^*}\mathbb{E}[\theta_{n,K}] + 1) \\ &\leq VC(2\sum_{n\neq a^*}\left[\frac{8\ln K}{\delta_n^2} + 1 + \frac{\pi^2}{3}\right] + 1). \end{aligned} \tag{55}$$

By adding SR and HR, we prove Proposition 1.

Appendix C
Proof of Theorem 2

Let $d_L^*(m)$ and $e_L^*(m)$ be the delay and energy consumption of task m achieved by EMM-LSI algorithm, respectively. From (21), we get

$$Vd_L^*(m) + q(m)e_L^* \leq Vd_G^*(m) + q(m)e_G^*(m) + W. \tag{56}$$

Substituting (56) into (38), we get

$$\Delta_J(rJ) + V_r\sum_{m=rJ+1}^{(r+1)J} d_L^*(m) \leq UJ^2 + V_rJg_r^* + WJ. \tag{57}$$

Thus

$$\begin{aligned} q((r+1)J+1) &= \sqrt{2\Delta_J(rJ)} \\ &\leq\sqrt{2[UJ^2 + V_rJg_r^* + WJ]}. \end{aligned} \tag{58}$$

By (39), we have

$$\sum_{m=rJ+1}^{(r+1)J} y(m) \leq \sqrt{2[UJ^2 + V_r J g_r^* + WJ]}. \tag{59}$$

By summing over $r = 0, 1, ..., R-1$ we prove part (2) of Theorem 2.

By dividing both sides of (57) by V_r, it follows that

$$\sum_{m=rJ+1}^{(r+1)J} d_L^*(m) \leq J g_r^* + \frac{UJ^2 + WJ}{V_r}. \tag{60}$$

Thus, by summing over $r = 0, 1, ..., R-1$ and dividing both sides by RJ, we prove part (1) of Theorem 2.

Appendix D
Proof of Proposition 2

We only focus on the regret in one task, and thus omit m for notation convenience. We first prove that both the sampling regret and hanover regret in each epoch is $O(\ln K)$.

We first bound the expectation of θ_{n,b,K_b}, which indicates the connection times to an suboptimal BS n in each epoch b after offloading K_b tasks. Let l be any positive integer, $c_{k,s,u} = \sqrt{2\ln(k-u)/s}$. Let $z' = z/\beta$, and a^* be replaced by a_b^*, we have

$$\begin{aligned}\theta_{n,b,K_b} &= \sum_{k=K_{b-1}+1}^{K_b} \mathbb{I}\{a^k = n\} \\ &\leq l + \sum_{k=K_{b-1}+1}^{K_b} \mathbb{I}\{a^k = n, \theta_{n,k-1,b} \geq l\} \\ &\leq l + \sum_{k=K_{b-1}+1}^{K_b} \mathbb{I}\{\max_{K_{b-1}<s<k} \bar{z}'_{a^*,s} - c_{k,s,u_{a^*}} \geq \\ &\quad \min_{K_{b-1}+l\leq s_n<k} \bar{z}'_{n,s_n} - c_{k,s_n,u_n}\} \\ &\leq l + \sum_{k=K_{b-1}+1}^{K_b} \sum_{s=K_{b-1}+1}^{k-1} \sum_{s_n=K_{b-1}+l}^{k-1} \mathbb{I}\{\bar{z}'_{a^*,s} - c_{k,s,u_{a^*}} \\ &\quad \geq \bar{z}'_{n,s_n} - c_{k,s_n,u_n}\}.\end{aligned} \tag{61}$$

$\mathbb{I}\{\bar{z}'_{a^*,s} - c_{k,s,u_{a^*}} \geq \bar{z}'_{n,s_n} - c_{k,s_n,u_n}\}$ implies that at least one of the following three equations hold

$$\bar{z}'_{a^*,s} \geq Z(a^*)/\beta K + c_{k,s,u_{a^*}}, \tag{62}$$

$$\bar{z}'_{n,s_n} \leq Z(n)/\beta K - c_{k,s_n,u_n}, \tag{63}$$

$$Z(a^*)/\beta K > Z(n)/\beta K - 2c_{k,s_n,u_n}. \tag{64}$$

By using Chernoff-Hoeffding bound, we have

$$\mathbb{P}\{\bar{z}'_{a^*,s} \geq Z(a^*)/\beta K + c_{k,s,u_{a^*}}\} \leq (k-u_{a^*})^{-4}, \tag{65}$$

$$\mathbb{P}\{\bar{z}'_{n,s_n} \leq Z(n)/\beta K - c_{k,s_n,u_n}\} \leq (k-u_n)^{-4}. \tag{66}$$

When $l \geq \lceil \frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} \rceil$, (64) not holds because

$$\begin{aligned}&Z(a^*)\beta K - Z(n)\beta K + 2c_{k,s_n,u_n} \\ &\leq Z(a^*)\beta K - Z(n)\beta K + \delta_{n,b} = 0,\end{aligned} \tag{67}$$

where $\delta_{n,b} = (Z(n) - Z(a_{m,b}^*))/K\beta$.

Then for any $n \neq a^*$, we have

$$\begin{aligned}&\mathbb{E}[\theta_{n,b,K_b}] \leq \\ &\lceil \frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} \rceil + \sum_{k=1}^{\infty} \sum_{s=K_{b-1}+1}^{k-1} \sum_{s_n=K_{b-1}+l}^{k-1} (\mathbb{P}\{\bar{z}'_{a^*,s} \geq \\ &Z(a^*)/\beta K + c_{k,s,u_{a^*}}\} + \mathbb{P}\{\bar{z}'_{n,s_n} \leq Z(n)/\beta K - c_{k,s_n,u_n}\}) \\ &\leq \lceil \frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} \rceil + \sum_{k=K_{b-1}+1}^{K_b} \sum_{s=K_{b-1}+1}^{k-1} \sum_{s_n=K_{b-1}+l}^{k-1} \\ &((k-u_{a^*})^{-4} + (k-u_n)^{-4}) \\ &\leq \frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} + 1 + \sum_{k=K_{b-1}+1}^{K_b} ((k-u_{a^*})^{-2} + (k-u_n)^{-2}) \\ &\leq \frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} + 1 + \sum_{k=1}^{\infty} 2k^{-2} \\ &\leq \frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} + 1 + \frac{\pi^2}{3}.\end{aligned} \tag{68}$$

Since the sampling regret in epoch b is $\text{SR}'_b = \sum_{n\neq a_b^*} \beta\delta_{n,b}\mathbb{E}[\theta_{n,b,K_b}]$, SR'_b is $O(\ln K)$.

The handover regret

$$\begin{aligned}\text{HR}'_b &= VC \sum_{n\in\mathcal{A}_{m,b}} \mathbb{E}[S_{m,n,b}] \\ &\leq C(2\sum_{n\neq a_b^*} \mathbb{E}[\theta_{n,b,K_b}] + 1) \\ &\leq C(2\sum_{n\neq a_b^*} \left[\frac{8\ln(K_b-u_n)}{\delta_{n,b}^2} + 1 + \frac{\pi^2}{3}\right] + 1).\end{aligned} \tag{69}$$

Thus the handover regret HR'_b is also $O(\ln K)$.

By summing SR'_b and HR'_b, and consider totally B epochs, the total regret for each task is $O(B\ln K)$.