Service Placement Using Distributed Pure Exploration in Linear Bandits

Mariam Yahya * †, Aydin Sezgin †, and Setareh Maghsudi†
* Department of Computer Science, University of Tübingen, Germany
† Faculty of Electrical Engineering and Information Technology, Ruhr-University Bochum, Germany

***Abstract*—The service placement problem in mobile edge computing focuses on determining which services to deploy at the network edge to maximize the quality of service. This problem is challenging because of the unknown service demand and network uncertainty. In this paper, we model the service demand as a linear function of the service attributes and formulate the problem as a linear bandit. We propose a novel distributed and adaptive multi-agent best arm identification algorithm under the fixed-confidence setting for linear bandits. The numerical results show that our algorithm solves the service placement problem with high confidence and leverages agent collaboration to achieve a near-optimal speedup. We also derive the theoretical upper bound on the sample complexity and communication overhead.**

***Index Terms*—Best arm identification (BAI), collaborative learning, linear bandits, service placement**

I. Introduction

As mobile communication technologies advance rapidly and the number of 5G users is expected to reach 6.3 billion by 2030 [1], the development of innovative infrastructure becomes a pressing necessity. Cloud computing enables devices to offload complex tasks to the cloud, which offers substantial computing and storage resources to support complex applications. However, this introduces high latency and security concerns. Multi-access edge computing (MEC) brings computation and storage resources closer to end users, reducing latency and improving security and quality of service (QoS) [2]. Nevertheless, dynamic user behavior and uncertainties in the environment pose significant challenges in meeting the growing demand for edge services.

The service placement (caching) problem addresses the challenge of determining which services to place at the resource-limited network edges to optimize performance metrics such as QoS. Services not placed at the edge are requested from the cloud or other edge servers [3], [4]. Since service demand is usually unknown, multi-armed bandit (MAB) approaches have been applied to learn demand online. MAB is a sequential decision-making framework where, in each round, the learner selects one of K arms and receives a reward drawn from a fixed but unknown distribution [5]. Contextual MAB is a variant that leverages contextual information, characterizing the service or the network [4], [6], to enhance learning.

In this paper, we propose a distributed and adaptive best arm identification (BAI) algorithm for multi-agent linear bandits (LB) in the fixed-confidence setting to optimize service placement at the network edge. Unlike conventional MABs that adapt service selection to real-time requests, BAI focuses on identifying the best service for long-term deployment.

We model service placement as a distributed multi-agent BAI problem, where edge servers mounted on small base stations (SBS) act as agents and the set of services corresponds to the arms. The SBSs collaborate to find the best placement by sharing observations through a central coordinator, achieving near-optimal speedup M, where M is the number of SBSs. Contextual information on long-term average service popularity is used to enhance learning.

BAI strategies in LBs differ from classical MABs due to arm correlations. In LBs, pulling suboptimal arms can still improve the estimation of the underlying linear model. Soare et al. [7] were the first to address the BAI problem in LB under a fixed-confidence setting. They proposed the $\mathcal{XY}$-Static algorithm, which uses optimal experimental design for arm selection but suffers from high sample complexity. To enhance efficiency, they introduced the $\mathcal{XY}$-Adaptive algorithm, which operates in phases, applying static allocation and eliminating less informative samples. In contrast, Xu et al. [8] proposed the Linear Gap-based Exploration (LinGapE), a fully adaptive algorithm that adjusts arm selection based on past observations.

Our proposed algorithm, named Distributed LinGapE (DistLinGapE) extends the work in [8] to a collaborative multi-agent case. In short, the main contributions of this paper are:

- We propose a collaborative and adaptive multi-agent algorithm for BAI in the LB fixed-confidence framework.
- We formulate the service placement problem in small cell networks as a LB problem, aiming to identify the optimal service placement that maximizes the reduction in the total user-perceived delay.
- To the best of our knowledge, this is the first work to apply the BAI setting in LB to MEC in general, including the service placement problem. We use our algorithm to find the best service placement while reducing the learning rounds per SBS compared to independent learning.
- We obtain the theoretical upper bounds on the number communication rounds and the sample complexity.
- Numerical results show the effectiveness of our algorithm in finding the best service placement and achieving a near-optimal speedup.

II. System Model

We first introduce the notation used throughout this work. **Notations:** We use boldface lowercase and boldface uppercase letters to represent vectors and matrices, respectively. We use

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany under grants 16KISK037 and 16KISK035.

$[K]$ to denote the set $\{1, \ldots, K\}$. Besides, $\|\boldsymbol{x}\|_p$ denotes the $p-$norm of a vector $\boldsymbol{x}$, and the weighted $2-$norm of vector $\boldsymbol{x}$ is defined by $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}$. Furthermore, we use $\mathbb{P}[\cdot]$ to denote the probability of the expression inside the brackets.

We consider a system with a macro base station (MBS) connected to the cloud and M homogeneous SBSs, each providing communication coverage and services to users within M non-overlapping coverage areas of similar size. The storage and computational resources at the cloud are large and can host all services, whereas the resources at the SBSs are significantly limited. The MBS provides wide-area coverage and forwards user service requests to the cloud. The SBSs are connected to the MBS in a star-shaped network topology to facilitate information exchange.

A service provider manages K services that can be deployed either at the cloud or the SBSs. Each service $k \in [K]$ is characterized by a d-dimensional context vector $\boldsymbol{x}_k \in \mathbb{R}^d$, representing various factors such as the service type, computational and memory requirements, and geographic or time-based demand. Here, our optimization problem depends on the service demand, so we assume that $\boldsymbol{x}_k$ is the long-term average service demand over d time periods. Specifically, $\boldsymbol{x}_k$ is an 8-dimensional vector representing the users' average demand for service k over a 24-hour period, sampled every three hours.

Due to limited edge resources, each SBS $m \in [M]$ can host only one service at any given time. If the service requested by the user is unavailable at the SBS, it has to offload the request to the cloud via the MBS, incurring additional delay. In this work, we aim to identify the service that minimizes the total users' delay the most when placed at the SBSs instead of the cloud. This problem is challenging because the service demand is unknown a priori.

The service placement process operates in discrete time steps indexed by $t = 1, 2, \ldots$. At each t, each SBS selects a service $a_{t,m} \in [K]$ with vector $\boldsymbol{x}_{a_{t,m}}$ and observes the resulting delay. To facilitate learning the service demand and effectively model the relationship between the context and the resulting demand, we adopt the common approach of representing service demand as a noisy linear function of the context [9]. Mathematically, the demand for service $a_{t,m}$ is

$$p_{t,m} = \boldsymbol{x}_{a_{t,m}}^\top \boldsymbol{\omega} + \xi_{t,m}, \tag{1}$$

where $\boldsymbol{\omega} \in \mathbb{R}^d$ is an unknown parameter vector representing the underlying relationship between services' contexts and demands, $\xi_{t,m}$ is the noise. User delay varies based on service placement at SBSs or the cloud, as explained below.

Service Delay at an SBS: If a service is placed at an SBS, the service delay consists of the transmission delay of tasks from the users to the SBS and the processing delay at the SBS. We assume that the task response delay is negligible [6]. For the uplink delay, we use the average data rate across all users because we are concerned with long-time deployment, and the instantaneous user data rates are unknown. The uplink transmission rate to an SBS, averaged across the users, is given by $\rho_{t,m} = W \log\left(1 + \frac{P h_{t,m}}{I + \sigma_{\mathrm{N}}^2}\right)$, where W is the bandwidth of the uplink channel, P is the user's transmission power, $h_{t,m}$ is the channel gain at time t, I is the interference power, and σ_{N}^2 is the noise power. Let s_k be the size of task k. Then, the communication delay of task k at SBS m is given by $\frac{s_k}{\rho_{t,m}}$. The round trip delay is $RTT_{t,m}$. The processing delay at SBS m is given by $\frac{cs_k}{f_{t,k,m}}$, where $f_{t,k,m}$ is the CPU capacity assigned to service k at SBS m and time t, and c is the number of CPU cycles required per bit. Thus, the total delay for service k at SBS m at time t is

$$d_{t,m}(k) = \frac{s_k}{\rho_{t,m}} + RTT_{t,m} + \frac{cs_k}{f_{t,k,m}}. \tag{2}$$

Service Delay at the Cloud: If the service requested by users in the service area of SBS m is not hosted at SBS m, the task request is offloaded to the cloud via the MBS. Let 0 denote the cloud index, and let $\rho_{t,0}$ be the average data rate between the users and the MBS. Then, the wireless transmission delay is $\frac{s_k}{\rho_{t,0}}$. The transmission delay in the backbone network with data rate ρ_t^{b} is $\frac{s_k}{\rho_t^{\mathrm{b}}}$. The round-trip delay is $RTT_{t,0}$. Additionally, the processing delay at the cloud is $\frac{cs_k}{f_{t,k,0}}$, where $f_{t,k,0}$ is the CPU capacity for service k at time t. Thus, the total delay is

$$d_{t,0}(k) = \frac{s_k}{\rho_{t,0}} + \frac{s_k}{\rho_t^{\mathrm{b}}} + RTT_{t,0} + \frac{cs_k}{f_{t,k,0}}. \tag{3}$$

Service Placement Utility: The utility of a service placement is the improvement in the total users' delay resulting from placing the service at the SBSs instead of the cloud. Let $a_{t,m} \in [K]$ be the service placed at SBS m at time t. The utility of this placement for one task is $d_{t,0}(a_{t,m}) - d_{t,m}(a_{t,m})$. The utility for the service demand given in (1) is $(d_{t,0}(a_{t,m}) - d_{t,m}(a_{t,m}))\, p_{t,m}$. This can be expressed as

$$r_{t,m}(a_{t,m}) = \boldsymbol{x}_{a_{t,m}}^\top \boldsymbol{\theta}^* + \eta_{t,m}, \tag{4}$$

where $\boldsymbol{\theta}^* = (d_{t,0}(a_{t,m}) - d_{t,m}(a_{t,m}))\, \boldsymbol{\omega}$, and $\eta_{t,m}$ is an independent, zero-mean, R-sub-Gaussian noise variable.

III. Problem Formulation

Let $a^* = \arg\max_{a \in [K]} \boldsymbol{x}_a^\top \boldsymbol{\theta}^*$ denote the optimal service that maximizes the expected utility. We aim to identify the estimated best service $\hat{a}_m^*$ such that

$$\mathbb{P}\left[(\boldsymbol{x}_{a^*} - \boldsymbol{x}_{\hat{a}_m^*})^\top \boldsymbol{\theta}^* > \epsilon\right] \leq \delta \tag{5}$$

as fast as possible. Here, ϵ is the desired accuracy and δ is the confidence. [1] Since all SBSs access the same set of services and share the same vector $\boldsymbol{\theta}^*$, there is a single optimal service for all SBSs, i.e., $\hat{a}^* = \hat{a}_m^*$.

IV. Modeling Service Placement as a Distributed BAI Problem

We model the service placement problem as a collaborative BAI problem and solve it using the DistLinGapE algorithm, presented in Section V. Each SBS $m \in [M]$ acts as an agent, while each service $k \in [K]$ corresponds to an arm associated with a context vector $\boldsymbol{x}_k \in \mathbb{R}^d$. The same set of K services is available to all SBSs. In this setting, selecting an arm in the bandit framework corresponds to an SBS m hosting a service.

[1] Since we aim to identify the best arm (service), we assume that $\epsilon = 0$, but the work remains valid as an (ϵ, δ)-PAC algorithm where $\epsilon > 0$.

Observing the reward at time t involves observing the total reduction in delay resulting from this service placement among all users in the SBS's coverage area.

Our problem can be solved by iteratively learning the parameter vector $\boldsymbol{\theta}^*$ using the BAI algorithm. Since the SBSs share the same vector $\boldsymbol{\theta}^*$ and can communicate through the MBS, collaboration in identifying the best service speeds up the learning process significantly. At a high level, the process of identifying the best service placement proceeds as follows. At time t, each SBS m hosts a service $a_{t,m}$ and observes the corresponding reward (utility). This reward is used to update the estimate of the unknown parameter vector, denoted $\hat{\boldsymbol{\theta}}_{t,m}$, which guides the next service placement decision. To minimize communication overhead, each SBS continues hosting services until a significant change in information is detected at one of the SBSs. When such a change occurs, a communication round is triggered, and the SBSs share their observations. This process repeats until the best service is identified. Section V describes the proposed algorithm in detail.

V. The DistLinGapE Algorithm

We consider a distributed LB problem with M agents, where $m \in [M]$, that exchange updates through a central coordinator. All agents have access to the same set of K arms, where each arm $k \in [K]$ is associated with a d-dimensional context vector $\boldsymbol{x}_k$, with $\|\boldsymbol{x}_k\| \leq L$ for some constant L. In round t, agent m pulls arm $a_{t,m} \in [K]$ and receives an immediate reward given by (4), where $\|\boldsymbol{\theta}^*\| \leq S$. The algorithm terminates when any agent identifies the optimal arm, which is then shared among all agents as the global best arm. The proposed DistLinGapE algorithm is detailed in Algorithm 1 on the next page. The BAI process involves sequentially deciding which arm to pull and constructing the confidence set based on the observed reward until the best arm is identified [7], [8].

A. Construction of Confidence Sets

Each agent uses the information available at time t to find the ℓ_2-regularized least squares estimate of $\boldsymbol{\theta}^*$, i.e., $\hat{\boldsymbol{\theta}}_{t,m}$. This information includes the locally updated $d \times d$ matrix given by $\Delta \boldsymbol{A}_{t,m} = \sum_{s=t_n+1}^{t} \boldsymbol{x}_{a_s,m} \boldsymbol{x}_{a_s,m}^\top$, where t_n is the index of the n-th communication round, and $\Delta \boldsymbol{b}_{t,m} = \sum_{s=t_n+1}^{t} \boldsymbol{x}_{a_s,m} r_{s,m}$. The agents share $\Delta \boldsymbol{A}_{t,m}$ and $\Delta \boldsymbol{b}_{t,m}$ with the coordinator in each communication round. The coordinator aggregates this information to obtain $\boldsymbol{A}_{\text{coor},t}$ and $\boldsymbol{b}_{\text{coor},t}$, and shares it with the agents for use in the next communication round. Consequently, at time t each agent has access to the coordinator's data and the locally collected data, as given in line 7 of Algorithm 1. With this information, the estimated parameter vector is given by $\hat{\boldsymbol{\theta}}_{t,m} = \boldsymbol{A}_{t,m}^{-1} \boldsymbol{b}_{t,m}$, with the confidence bound given in Proposition 1.

Proposition 1. ***Confidence Ellipsoid*** *[10, Th. 2] For LBs with conditionally R-sub-Gaussian noise, for $R \geq 0$, let $\|\boldsymbol{\theta}^*\|_2 \leq S$, and λ be the regularization parameter. Then, the relation*

$$|\boldsymbol{x}^\top (\hat{\boldsymbol{\theta}}_{t,m} - \boldsymbol{\theta}^*)| \leq \|\boldsymbol{x}\|_{\boldsymbol{A}_{t,m}^{-1}} C_{t,m} \tag{6}$$

holds for $t \in \{1, 2, \dots\}$, with confidence $\delta = M\delta_m$, where

$$C_{t,m} = R\sqrt{2 \log \frac{\det(\boldsymbol{A}_{t,m})^{\frac{1}{2}}}{\lambda^{\frac{d}{2}} \delta_m}} + \lambda^{\frac{1}{2}} S. \tag{7}$$

B. The Arm Selection Strategy

In each round t, each agent m uses $\boldsymbol{A}_{t,m}$ to find the two arms whose gap needs to be estimated: the arm with the highest estimated reward $i_{t,m}$, and the most ambiguous arm $j_{t,m}$, given in lines 24 and 25 of Algorithm 1, respectively. Afterwards, it pulls the arm with the largest information gain in estimating the expected reward gap: $\hat{\Delta}_{t,m}(i,j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \hat{\boldsymbol{\theta}}_{t,m}$ with confidence $\beta_{t,m}(i,j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{A}_{t,m}^{-1}} C_{t,m}$. In each round t, the agents check if the stopping condition is met, that is, if the upper confidence bound on the gap of the expected reward falls below the target accuracy ϵ. Formally,

$$B_m(t) = \hat{\Delta}_{t,m}(j_{t,m}, i_{t,m}) + \beta_{t,m}(j_{t,m}, i_{t,m}) \leq \epsilon. \tag{8}$$

If the condition is satisfied, the algorithm stops and the arm with the highest estimated reward is identified as the best arm. Otherwise, the algorithm proceeds to pull the next arm.

Following [7], [8], there are two approaches for the arm selection strategy. In the greedy approach, each agent pulls the arm that minimizes the confidence bound, given by

$$a_{t+1,m} = \arg\min_{a \in [K]} \|\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}}\|_{(\boldsymbol{A}_{t,m} + \boldsymbol{x}_a \boldsymbol{x}_a^\top)^{-1}}. \tag{9}$$

Although we do not analyze the theoretical guarantees of this approach, we empirically show that it performs well in the numerical results [8]. Alternatively, the second approach involves finding the optimal ratio for arm k appearing in the optimal allocation sequence at agent m as $t \to \infty$ [8]. This ratio minimizes $\|\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}}\|_{\boldsymbol{A}_{t,m}^{-1}}$. Then, the arm achieving this ratio as closely as possible is selected. Let $w_k^*(i_{t,m}, j_{t,m})$ be the k-th element in $\mathbf{w}^*(i_{t,m}, j_{t,m})$ defined as $\mathbf{w}^*(i_{t,m}, j_{t,m}) = \arg\min_{\mathbf{w} \in \mathbb{R}^d} |\mathbf{w}|$, such that $\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}} = \sum_{k=1}^{K} w_k \boldsymbol{x}_k$. The optimal ratio for selecting arm k is [8]

$$p_k^*(i_{t,m}, j_{t,m}) = \frac{|w_k^*(i_{t,m}, j_{t,m})|}{\sum_{k=1}^{K} |w_k^*(i_{t,m}, j_{t,m})|}. \tag{10}$$

Let $T_{a,m}(t)$ be the number of times arm a is selected up to time t that is available to agent m. The arm selection strategy aims to keep the actual arm selection ratio as close as possible to the target ratio. Thus, the pulled arm satisfies

$$a_{t+1,m} = \arg\min_{a \in [K]: p_k^*(i_{t,m}, j_{t,m}) > 0} \frac{T_{a,m}(t)}{p_k^*(i_{t,m}, j_{t,m})}. \tag{11}$$

In the DistLinGapE, collaboration between agents reduces their individual sample complexity. This reduction is measured by the speedup over a centralized algorithm, defined as $S_{\mathcal{A}} = \frac{T_{\mathcal{O}}}{T_{\mathcal{A}}}$, where $T_{\mathcal{O}}$ and $T_{\mathcal{A}}$ are the per-agent sample complexities of the best centralized and collaborative algorithms, respectively [11]. The speedup satisfies $S_{\mathcal{A}} \leq M$, where M is the number of agents and represents the maximum speedup.

The upper bound on sample complexity for the arm selection strategy in (11) is given in Theorem 1. We first define the

problem complexity $H_\epsilon = \sum_{k=1}^{K} \max_{i,j \in [K]} \frac{p_k^*(i,j)\alpha(i,j)}{\max\left(\epsilon, \frac{\epsilon+\Delta_i}{3}, \frac{\epsilon+\Delta_j}{3}\right)^2}$, where $\alpha(i,j) = |\mathbf{w}^*(i,j)|$ and $\Delta_i = (\boldsymbol{x}_{a^*} - \boldsymbol{x}_i)^\top \boldsymbol{\theta}^*$ [8]. We omit the proof due to length limitations.

Theorem 1. *The DistLinGapE algorithm can identify the $(\epsilon, M\delta_m)$-best arm using the arm selection strategy in* (9) *or* (11) *with probability $1 - M\delta_m$ and the following sample complexity per agent: For $\lambda \leq \frac{2R^2}{S^2} \log \frac{K^2}{\delta_m}$: $\tau_m \leq \mu + \frac{4H_\epsilon R^2}{M}\left(2\log\frac{K^2}{\delta_m} + d\log\left(1 + \frac{Y^2L^2}{\lambda d}\right)\right)$, where $\mu = \frac{K}{M} + 1$, $Y = 2\sqrt{16H_\epsilon^2 R^4 dL^2/(M\lambda) + N^2}$, and $N = \frac{8H_\epsilon R^2}{M}\log\frac{k^2}{\delta_m}$. For $\lambda > 4H_\epsilon R^2 L^2$: $\tau_m \leq 2\left(\frac{4H_\epsilon R^2}{M}\log\frac{K^2}{\delta_m} + \frac{2H_\epsilon \lambda S^2}{M} + \mu\right)$.*

C. Communication Rounds

To reduce the communication overhead, an agent m initiates a communication round only when there is a significant change in the matrix $\boldsymbol{A}_{t,m}$. This change is evaluated by the ratio $\frac{\det(\boldsymbol{A}_{t,m})}{\det(\lambda \boldsymbol{I} + \boldsymbol{A}_{\text{coor},t})}$ given in line 14 of Algorithm 1, where $\det(\cdot)$ is the determinant and $\boldsymbol{I}$ is the identity matrix [12]. The threshold D controls the communication frequency. Let τ be the total sample complexity. The upper bound on the number of communication rounds is given by Theorem 2, we omit the proof due to the space limit.

Theorem 2. *The total communication cost of the DistLinGapE algorithm is upper bounded by $O\left(\sqrt{\frac{M\tau d \log_2 \tau}{D}}\right)$.*

VI. Numerical Results

We compare the performance of DistLinGapE with (a) the $\mathcal{XY}$-Oracle [7], which assumes knowledge of $\boldsymbol{\theta}^*$, (b) the $\mathcal{XY}$-Adaptive [7], which runs in phases and eliminates uninformative directions after each phase, and (c) LinGapE [8], a fully adaptive BAI algorithm for the single-agent setting. Results are averaged over 30 runs. Despite setting each agent's confidence to $\delta_m = 0.05$, DistLinGapE consistently identifies the optimal arm with zero errors.

A. Simulation on Synthetic Data

We use the benchmark example from [7], commonly used in BAI in LB. This example consists of $d+1$ arms. The context vectors of the first d arms are the canonical basis vectors, given by $\boldsymbol{x}_1 = \boldsymbol{e}_1, \ldots, \boldsymbol{x}_d = \boldsymbol{e}_d$. The context vector of the last arm is $\boldsymbol{x}_{d+1} = [\cos(\phi), \sin(\phi), 0, \ldots, 0]^\top$ with $\phi = 0.01$. The parameter vector is $\boldsymbol{\theta}^* = [2, 0, \ldots, 0]^\top$ and the noise follows $\eta_{t,m} \sim \mathcal{N}(0,1)$. Since $\arg\max_{k \in [K]} \boldsymbol{x}_k^\top \boldsymbol{\theta}^* = 1$, arm 1 is the optimal arm a^* in this example.

Fig. 1 shows how the sample complexity changes with the dimension d. DistLinGapE, with $M = 4$ agents, achieves a total sample complexity close to LinGapE, effectively reaching the optimal speedup of M. Since agents pull arms sequentially, each agent's sample complexity is $1/M$ of the total, as shown by the dashed line. The semi-adaptive $\mathcal{XY}$-Adaptive algorithm requires more samples to reach the same confidence. Despite knowing $\boldsymbol{\theta}^*$, $\mathcal{XY}$-Oracle has a higher sample complexity than DistLinGapE because DistLinGapE has tighter confidence bounds, which enable more accurate reward estimates [8].

Algorithm 1 The DistLinGapE Algorithm

1: **Input:** ϵ, δ, S, R, and λ
2: **Output** Arm $\hat{a}^*$ that satisfies stopping condition.
3: **Initialize agents**: $\forall m \in [M]$, $\boldsymbol{A}_{0,m} = \boldsymbol{0}$, $\boldsymbol{b}_{0,m} = \boldsymbol{0}$, $\Delta\boldsymbol{A}_{0,m} = \boldsymbol{0}$, $\Delta\boldsymbol{b}_{0,m} = \boldsymbol{0}$, $\Delta t_{0,m} = 0$; $\forall k \in [K]$, $\Delta T_{k,m} = 0$
4: **Initialize coordinator**: $\boldsymbol{A}_{\text{coor},0} = \boldsymbol{0}$, $\boldsymbol{b}_{\text{coor},0} = \boldsymbol{0}$, $T_k = 0$
5: **for** $t = 1, \ldots, \infty$ **do**
6: **for** Agent $m = 1, \ldots, M$ **do**
7: $\boldsymbol{A}_{t,m} = \lambda \boldsymbol{I} + \boldsymbol{A}_{\text{coor},t} + \Delta\boldsymbol{A}_{t,m}$, $\boldsymbol{b}_{t,m} = \boldsymbol{b}_{\text{coor},t} + \Delta\boldsymbol{b}_{t,m}$, and $\Delta t_{t,m} += 1$
8: Select pulling direction: $(i_{t,m}, j_{t,m}, B_m(t)) \leftarrow$ SELECT-DIRECTION$(\boldsymbol{A}_{t,m}, \boldsymbol{b}_{t,m})$
9: **if** $B_m(t) \leq \epsilon$ **then**
10: $i_{t,m}$ is the best arm $\hat{a}^*$; Terminate algorithm
11: **end if**
12: Pull $a_{t,m}$ according to (9) or (11) and observe $r_{t,m}$
13: Update $\Delta\ \boldsymbol{A}_{t,m} += \boldsymbol{x}_{a_{t,m}} \boldsymbol{x}_{a_{t,m}}^\top$, $\Delta\ \boldsymbol{b}_{t,m} += \boldsymbol{x}_{a_{t,m}} r_{t,m}$, $\Delta\ T_{a_{t,m},m}(t) += 1$, $T_{a_{t,m},m}(t) = T_{a_{t,m}}(t) + \Delta\ T_{a_{t,m},m}(t)$
// Communication condition
14: **if** $\Delta t_{t,m} \log \frac{\det(\boldsymbol{A}_{t,m})}{\det(\lambda \boldsymbol{I} + \boldsymbol{A}_{\text{coor},t})} > D$ **then**
15: Each agent $m \in [M]$ sends $\Delta\boldsymbol{A}_{t,m}, \Delta\boldsymbol{b}_{t,m}$, and $\Delta T_{k,m}(t), \forall k \in [K]$ to coordinator and resets $\Delta\boldsymbol{A}_{t,m} = 0, \Delta\boldsymbol{b}_{t,m} = 0$, $\Delta t_{t,m} = 0$, $\Delta T_{k,m} = 0$.
16: Coordinator collects $\Delta\boldsymbol{A}_{t,m}$, $\Delta\boldsymbol{b}_{t,m}, \Delta T_{k,m}(t)$
17: Coordinator computes $\boldsymbol{A}_{\text{coor},t} += \sum_{m=1}^{M} \Delta\boldsymbol{A}_{t,m}$, $\boldsymbol{b}_{\text{coor},t} += \sum_{m=1}^{M} \Delta\boldsymbol{b}_{t,m}, T_k(t) += \sum_{m=1}^{M} \Delta T_{k,m}(t)$
18: Coordinator broadcasts $\boldsymbol{A}_{\text{coor},t}, \boldsymbol{b}_{\text{coor},t}, T_k(t)$ to agents
19: **end if**
20: **end for**
21: **end for**
22: **Procedure** Select-Direction$(\boldsymbol{A}_{t,m}, \boldsymbol{b}_{t,m})$
23: $\hat{\boldsymbol{\theta}}_{t,m} \leftarrow \boldsymbol{A}_{t,m}^{-1} \boldsymbol{b}_{t,m}$
24: $i_{t,m} \leftarrow \arg\max_{i \in [K]} (\boldsymbol{x}_i^\top \hat{\boldsymbol{\theta}}_{t,m})$
25: $j_{t,m} \leftarrow \arg\max_{j \in [K]} (\hat{\Delta}_{t,m}(j, i_{t,m}) + \beta_{t,m}(j, i_{t,m}))$
26: $B_m(t) \leftarrow \max_{j \in [K]} (\hat{\Delta}_{t,m}(j, i_{t,m}) + \beta_{t,m}(j, i_{t,m}))$
27: **return** $i_{t,m}, j_{t,m}, B_m(t)$
28: **EndProcedure**

TABLE I: Number of Samples per Arm for $d = 5$.

Arm	DistLinGapE	LinGapE	$\mathcal{XY}$-Adaptive	$\mathcal{XY}$-Oracle
Arm 1	794.36	727.77	3898.20	1623.73
Arm 2	153060.47	147117.53	776540.00	340027.27
Arm 3	34.07	27.10	30.27	1618.00
Arm 4	40.43	25.47	18.30	1620.10
Arm 5	33.10	28.90	20.73	1604.80
Arm 6	8.47	4.97	20.53	1616.67

Table I shows the number of samples per arm for $d = 5$. The DistLinGapE column represents the sum of pulls across all 4 agents. Arm 2 is pulled significantly more than any other arm, including the optimal arm 1. This is because, for small values of ϕ, $0 < \phi \ll 1$, arm $d+1$ is the strongest competitor to arm 1, with $\Delta_{\min} = (\boldsymbol{x}_1 - \boldsymbol{x}_{d+1})^\top \boldsymbol{\theta}^*$. To accurately identify arm 1 as the best, the uncertainty in the direction $\boldsymbol{y} = (\boldsymbol{x}_1 - \boldsymbol{x}_{d+1})$ must be minimized. Since arm 2 is nearly aligned with this

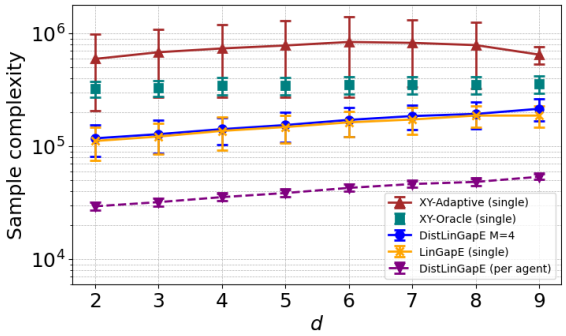
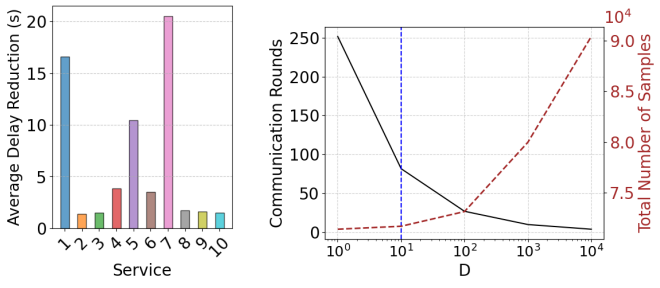


Fig. 1: The sample complexity for different values of d.

direction, it is the most informative choice. The number of samples per agent in DistLinGapE is $1/M$ of the table value.

B. Simulation on the Small Cell Network

Consider a network with a service provider with 10 services. The context of each service, $\boldsymbol{x}_k$, is an 8-dimensional vector representing the long-term average service demand over 24 hours, divided into eight three-hour periods. To model $\boldsymbol{x}_k$ in our simulation, the base demand for each service follows a Zipf distribution, with Gaussian noise added to capture demand fluctuations across different time periods. The channel bandwidth is 10 MHz, the transmission power is 10 dBm, and the noise and interference power are $\sigma_{\mathrm{N}}^2 = -104$ dBm and $I = -90$ dBm, respectively. The cloud CPU frequency is $f_{t,k,0} = [4.6, 5.6]$ GHz, while at each SBS, it is $f_{t,k,m} = [2.3, 3.2]$ GHz [6], with $c = 100$ cycles/bit. The task size is $s_k = [0.5, 1]$ MB, with bit rate $\rho_t^{\mathrm{b}} = [1, 4]$ Gbps. The round-trip time is $RRT_{t,0} = [20, 40]$ ms and $RRT_{t,m} = [2, 7]$ ms.



(a) The average delay reduction for each service k.

(b) The effect of the communication threshold D on performance.

Fig. 2: Performance of a network with $K = 10$ and $M = 4$.

Fig. 2a shows the average delay reduction from placing service k at the SBSs in a four-SBS network. Here, service 7 is optimal and the DistLinGapE algorithm correctly identifies it as the best with zero error despite the unknown service demand and randomness of the environment. Table II shows the speedup from collaborative learning when the threshold $D = 10$. The per-agent sample complexity for the centralized algorithm is $T_{\mathcal{O}} = 70513.15$ samples. If the SBSs learn independently, the total sample complexity would be $MT_{\mathcal{O}}$, but collaboration can achieve a speedup M. The sample complexity of the centralized $\mathcal{XY}$-Oracle algorithm is 657181.16 samples. It is higher than that of DistLinGapE because the latter has a tighter confidence bound.

TABLE II: Algorithm Speedup

M	1	2	4	6
samples/agent	70513.15	35440.81	17922.73	12078.15
$S_{\mathcal{A}}$	-	1.99	3.93	5.84

Fig. 2b shows the impact of the communication threshold D in a network with four SBSs. Small values of D lead to excessive communication overhead due to frequent information exchange, while large D increases sample complexity as each SBS performs more local updates before sharing information. Thus, carefully tuning D is important for balancing communication efficiency and sample complexity. The vertical blue line marks $D = 10$, which is used in Table II for $M = 4$.

VII. Conclusion and Future Work

This paper addresses optimal service placement in small cell networks with unknown service demand, aiming to maximize the reduction in total user delay when deploying a service in SBSs instead of the cloud. We model the service demand as a linear function of the long-term average demand and frame the problem as a distributed BAI problem. Our algorithm identifies the optimal service with high confidence and achieves near-optimal speedup. We also derive upper bounds on sample complexity and communication rounds.

References

[1] Ericsson, "Ericsson mobility report," (2024). Accessed: Jan. 2025. [Online] Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2024.

[2] Y. Li, L. Li, and Z. Zhou, "Joint edge caching and computation offloading for heterogeneous tasks in MEC-enabled vehicular networks," *Veh. Commun.*, vol. 50, p. 100860, 2024.

[3] W. Chu, X. Zhang, X. Jia, J. C. Lui, and Z. Wang, "Online optimal service caching for multi-access edge computing: A constrained multi-armed bandit optimization approach," *Comput. Netw.*, vol. 246, p. 110395, 2024.

[4] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE Conf. Comput. Commun. (INFOCOM)*, pp. 1468–1476, IEEE, 2019.

[5] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[6] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio–temporal edge service placement: A bandit learning approach," *IEEE Trans. on Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, 2018.

[7] M. Soare, A. Lazaric, and R. Munos, "Best-arm identification in linear bandits," *Adv. in Neural Inf. Proc. Syst.*, vol. 27, 2014.

[8] L. Xu, J. Honda, and M. Sugiyama, "A fully adaptive algorithm for pure exploration in linear bandits," in *Int. Conf. on Artif. Intell. and Statist.*, pp. 843–851, PMLR, 2018.

[9] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[10] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," *Adv. in Neural Inf. Proc. Syst.*, vol. 24, 2011.

[11] C. Tao, Q. Zhang, and Y. Zhou, "Collaborative learning with limited interaction: Tight bounds for distributed exploration in multi-armed bandits," in *IEEE 60th Annu. Symp. on Found. of Comput. Sci. (FOCS)*, pp. 126–146, IEEE, 2019.

[12] C. Li and H. Wang, "Communication efficient federated learning for generalized linear bandits," *Advances in Neural Inf. Process. Syst.*, vol. 35, pp. 38411–38423, 2022.