Bandit Learning for Resource Allocation and Performance Optimization in Edge Networks

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von
Mariam Yahya
aus Albira/Palästina

Tübingen
2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	16.12.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	Prof. Dr.-Ing. Setareh Maghsudi
2. Berichterstatter/-in:	Prof. Dr.-Ing. Falko Dressler

To my fellow students and researchers in Palestine,
those whose classrooms lie in ruins,
and those who struggle to reach them.

Abstract

Integrating AI-based methods into emerging 5G and 6G networks is essential for improving current network functions, unlocking new features, and accommodating the rapid increase in connected devices. In addition, these methods play a crucial role in supporting advanced applications such as smart cities, autonomous driving, and immersive experiences. A key technology that helps meet the stringent requirements of these applications is multi-access edge computing (MEC), which brings computational and storage resources closer to end users to enable low latency, high reliability, and enhanced privacy. However, MEC networks face significant uncertainty due to dynamic environments, unpredictable user availability and behavior, and limited resources. Motivated by these challenges, this thesis designs and applies bandit algorithms to address key decision-making problems in MEC networks under uncertainty. Additionally, it proposes methods to optimize network design for improved communication and learning performance, with a particular focus on federated learning (FL).

The first contribution addresses the service placement problem in small cell networks, where edge servers, modeled as bandit agents, aim to identify the best service to deploy locally in order to minimize total user delay compared to cloud deployment, under unknown service demand. To this end, we extend the best arm identification (BAI) algorithm for linear bandits in the fixed-confidence setting to a distributed and adaptive multi-agent scenario, where edge servers learn collaboratively. Numerical results show that the optimal service is identified with the desired confidence level and that collaboration accelerates learning in proportion to the number of participating servers. Additionally, we derive upper bounds on per-agent sample complexity and communication cost.

The second contribution focuses on decentralized task offloading and load balancing in dense MEC networks with random channel conditions and varying task sizes. Each user acts as an agent that offloads its task to a server where it can be completed within a specified time limit. Since users compete for shared communication and computational resources, their rewards depend on both their server selection and the actions of other users. We model this interaction as a mean-field multi-armed bandit game, where the environment appears stationary in dense networks. We then propose a load-balancing approach that adjusts users' rewards to influence their decisions and achieve a target load distribution across servers. We theoretically prove convergence to a steady-state load distribution and demonstrate our findings through numerical results.

The third contribution considers an unmanned aerial vehicle (UAV)-enabled network,

where UAVs provide sensor coverage and act as FL clients using data collected from energy-harvesting sensors. The goal is to determine optimal UAV coverage that jointly maximizes coverage and minimizes FL delay under uncertainty. After deriving the statistical distribution of the FL delay, the problem is formulated as a multi-objective BAI bandit problem. A general scalarization-based, cost-aware BAI algorithm is proposed to identify the optimal arms that maximize the ratio of expected reward to expected energy cost. We derive an upper bound on the algorithm's error probability, and numerical results demonstrate its varying effectiveness in identifying optimal arms under different parameter settings. The evaluation also compares the regret and fairness of algorithm variants, highlighting the trade-offs between competing objectives.

The final contribution addresses the problem of minimizing FL convergence time in UAV-enabled networks by jointly optimizing coverage and resource allocation. The proposed solution is divided into two stages. First, a heuristic method is designed to determine UAV placements that adjust coverage to reduce the computation time. Then, given that UAV locations influence communication delays, a fair channel allocation and power control strategy is employed to minimize communication time. These two stages collectively aim to mitigate the straggler effect in FL. Unlike previous work, this approach does not employ bandits, as the sensor distribution is assumed to be known. Simulation results demonstrate the effectiveness of the proposed joint optimization approach.

Overall, this work demonstrates the effectiveness of bandit algorithms in addressing key MEC challenges and motivates the development of new methods inspired by the problem setting. It also shows the critical role of efficient network design in supporting, and in some cases accelerating, the learning process.

Zusammenfassung

Die Integration KI-basierter Methoden in aufkommenden 5G- und 6G-Netzen ist entscheidend für die Verbesserung bestehender Netzwerkfunktionen, die Erschließung neuer Anwendungsmöglichkeiten und die Bewältigung des rasanten Anstiegs vernetzter Geräte. Darüber hinaus spielen diese Methoden eine zentrale Rolle bei der Unterstützung fortschrittlicher Anwendungen wie Smart Cities, autonomes Fahren und umfassender Erlebnisse. Eine Schlüsseltechnologie zur Erfüllung der hohen Anforderungen dieser Anwendungen ist das Multi-Access Edge Computing (MEC), das Rechen- und Speicherressourcen näher an die Endnutzer verlagert, um geringe Latenz, hohe Zuverlässigkeit und erhöhte Datensicherheit zu gewährleisten. MEC-Netzwerke sind jedoch mit erheblichen Unsicherheiten konfrontiert, die aus dynamischen Umgebungen, unvorhersehbarer Nutzerverfügbarkeit und begrenzten Ressourcen resultieren. Ausgehend von diesen Herausforderungen entwickelt und analysiert diese Dissertation Banditenalgorithmen zur Lösung zentraler Entscheidungsprobleme in MEC-Netzwerken unter Unsicherheit. Darüber hinaus werden Verfahren zur Optimierung des Netzwerkdesigns vorgestellt, um Kommunikation und Lernleistung zu verbessern, mit besonderem Fokus auf föderiertes Lernen (FL).

Der erste Beitrag behandelt das Problem der Dienstplatzierung in Small-Cell-Netzwerken, wobei Edge-Server als Banditenagenten modelliert werden, die das Ziel verfolgen, unter unbekannter Dienstnachfrage den optimalen Dienst lokal bereitzustellen, um die gesamte Nutzungsverzögerung im Vergleich zur Cloud-Verarbeitung zu minimieren. Zu diesem Zweck wird der Best-Arm-Identification (BAI)-Algorithmus für lineare Banditen im Fixed-Confidence-Setting auf ein verteiltes, adaptives Multi-Agenten-Szenario erweitert, in dem Edge-Server kollaborativ lernen. Numerische Ergebnisse zeigen, dass der optimale Dienst mit der gewünschten Wahrscheinlichkeit identifiziert wird und dass die Zusammenarbeit das Lernen proportional zur Anzahl der teilnehmenden Server beschleunigt. Zusätzlich werden obere Grenzen für die Stichprobenkomplexität pro Agent und die Kommunikationskosten bereitgestellt.

Der zweite Beitrag konzentriert sich auf dezentrale Aufgabenverteilung und Lastbalancierung in dichten MEC-Netzwerken mit zufälligen Kanalbedingungen und variierenden Aufgabenanforderungen. Jede Nutzerin bzw. jeder Nutzer agiert als Agent, der eine Aufgabe an einen Server auslagert, wo sie innerhalb eines vorgegebenen Zeitlimits verarbeitet werden soll. Da sich mehrere Nutzer die Kommunikations- und Rechenressourcen teilen, hängt die Belohnung eines einzelnen Nutzers sowohl von der Serverwahl

als auch von den Entscheidungen der anderen Nutzer ab. Dieses Zusammenspiel wird als Mean-Field-Multi-Armed-Bandit-Spiel modelliert, in dem die Umgebung bei hoher Nutzerdichte als stationär erscheint. Anschließend wird ein Lastverteilungsverfahren vorgeschlagen, das die Belohnungen so anpasst, sodass eine gezielte Lastverteilung über die Server hinweg erreicht wird. Die Konvergenz zu einer stabilen Lastverteilung wird theoretisch nachgewiesen und durch numerische Ergebnisse untermauert.

Der dritte Beitrag betrachtet ein von unbemannten Luftfahrzeugen (UAVs, unmanned aerial vehicles) unterstütztes Netzwerk, in dem UAVs Sensordatenerfassung leisten und als FL-Clients fungieren. Ziel ist es, eine UAV-Abdeckung zu bestimmen, die sowohl die Sensordatenabdeckung maximiert als auch die FL-Verzögerung unter Unsicherheit minimiert. Nach Herleitung der statistischen Verteilung der FL-Verzögerung wird das Problem als BAI-Banditenproblem mit multiplen Zielen formuliert. Es wird ein skalierungsbasiertes, kostenbewusstes BAI-Verfahren vorgeschlagen, das die Arme mit dem besten Verhältnis zwischen erwarteter Belohnung und erwarteten Energiekosten identifiziert. Eine obere Grenze für die Fehlerrate des Algorithmus wird hergeleitet. Simulationen zeigen die Effektivität des Verfahrens bei unterschiedlichen Parametereinstellungen. Zudem werden verschiedene Algorithmusvarianten in Bezug auf Regret und Fairness verglichen, wodurch Zielkonflikte zwischen konkurrierenden Kriterien aufgezeigt werden.

Der vierte Beitrag widmet sich der Minimierung der Konvergenzzeit von FL in UAV-gestützten Netzwerken durch die gemeinsame Optimierung von Netzwerkabdeckung und Ressourcenzuweisung. Der vorgeschlagene Lösungsansatz besteht aus zwei Phasen. In der ersten Phase wird ein heuristisches Verfahren entwickelt, das UAV-Positionen bestimmt, um die Rechenzeiten zu minimieren. Da die UAV-Positionen die Kommunikationsverzögerung beeinflussen, folgt in der zweiten Phase eine faire Kanalzuweisung und Leistungsregelung, um die Übertragungszeit zu minimieren. Gemeinsam zielen beide Phasen darauf ab, den sogenannten Straggler-Effekt im FL zu reduzieren. Im Gegensatz zu den vorherigen Beiträgen kommen hier keine Banditenmodelle zum Einsatz, da vorausgesetzt wird, dass die Sensorverteilung bekannt ist. Simulationen belegen die Effektivität der vorgeschlagenen Optimierungsmethode.

Insgesamt demonstriert diese Arbeit die Wirksamkeit von Banditenalgorithmen zur Bewältigung zentraler Herausforderungen in MEC-Netzwerken und motiviert die Entwicklung neuer, durch spezifische Probleme inspirierter Methoden. Darüber hinaus wird die entscheidende Rolle eines effizienten Netzwerkdesigns zur Unterstützung und in manchen Fällen zur Beschleunigung des Lernprozesses hervorgehoben.

Acknowledgments

I would like to start by sincerely thanking Prof. Dr.-Ing. Setareh Maghsudi for accepting me into the Decision Making group during her time at the University of Tübingen, and for her supervision and support throughout my PhD.

I would also like to sincerely thank my committee members: Prof. Dr.-Ing. Falko Dressler for kindly reviewing my thesis and for his invaluable feedback, Prof. Dr. Michael Menth and Prof. Dr. Claire Vernade for their time and insightful comments. I am also grateful to all committee members for their support in conducting the oral examination. Special thanks go to Claire for giving me the opportunity to attend her enriching reading groups.

Thank you to everyone who was once part of the Decision Making group for the valuable experience. To Behzad Nourani-Koliji, Ioannis Tsetis, Sofien Dhouib, Xiaotong Cheng, and Steven Bilaj, thank you for the many insightful scientific discussions, all the spontaneous and entertaining conversations, good times, and support. Although I spent less time with Michela Petriconi, Ahmad Ehyaei, Sabrine Chebbi, Qiang He, Haniyeh Barghi, Francesco Chini, Amir Balef, and Melodi Caliskan, I truly appreciate our time together.

I would like to acknowledge the German Academic Exchange Service (DAAD) for their financial support during the first four years of my PhD studies.

Living in beautiful Tübingen was a wonderful experience, made even more memorable by the amazing friends I met here. While I won't name them here, I am grateful for their presence, support, and for the many gatherings, coffees, walks, and ice creams in edible cups. Heartfelt thanks to my friends and travel companions in Germany, and to my friends back home.

Finally, to my family: thank you for your love and support, especially during moments of doubt. While I wished I could have been by your side during these past two difficult years, you were the ones looking after my well-being, giving me strength, and showing understanding when I couldn't travel home. Thanks for always being there, keeping our chats full of laughter, and making me feel close with every update. Buthaina and Adnan, I am deeply grateful to you.

Contents

List of Figures

List of Tables

Abbreviations

5G	Fifth generation
6G	Sixth generation
BAI	Best arm identification
BO-MAB	Bi-objective multi-armed bandits
CP	Circle Packing
Cheb-S	Chehychev scalarization
CMAB	Contextual multi-armed bandits
CPU	Central processing unit
DL	Downlink
FL	Federated learning
IHPPP	Inhomogeneous Poisson point process
IoT	Internet of things
LB	Linear bandit
Lin-S	Linear scalarization
MAB	Multi-armed bandit
MBS	Macro base station
MEC	Multi-access edge computing or mobile edge computing
MO-MAB	Multi-objective multi-armed bandits
MFG	Mean field game
QoS	Quality of service
SBS	Small base station
Seq-Halv	Sequential halving
Succ-Rej	Successive rejects
UL	Uplink
UCB	Upper confidence bound
UAV	Unmanned aerial vehicle
URLLC	Ultra-reliable low latency communications
WSN	Wireless sensor network

Notation

$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integers $\{\dots,-2,-1,0,1,2,\dots\}$
$\mathbb{Z}_+$	Set of positive integers $\{1,2,3,\dots\}$
x	Scalar
$\boldsymbol{x}$	Bold lowercase: vector
$\boldsymbol{X}$	Bold uppercase: matrix
$()^\top$	Transpose of a vector or matrix
$\det(\boldsymbol{X})$	Determinant of matrix $\boldsymbol{X}$
$\langle \boldsymbol{v}, \boldsymbol{u} \rangle$	Inner product: $\boldsymbol{v}^\top \boldsymbol{u}$
$\lvert \cdot \rvert$	L_1 norm (Absolute value)
$\lVert \cdot \rVert$	L_2 norm (Euclidean norm)
$\lVert \cdot \rVert_p$	L_p norm
$\lVert \boldsymbol{x} \rVert_{\boldsymbol{A}}$	Mahalanobis norm: $\sqrt{\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}}$
$\mathbb{E}[\cdot]$	Expectation
$\mathbb{P}[\cdot]$	Probability
$\operatorname{erf}(\cdot)$	Error function
$Q_m(a,b)$	The generalized Marcum Q-function
$[K]$	Set $\{1,2,\dots,K\}$
$\mathcal{X}$	A set
$\lvert \mathcal{X} \rvert$	Cardinality of set $\mathcal{X}$
$\mathbb{1}[\cdot]$	Indicator function, equals 1 if the condition inside brackets is true, and 0 otherwise
$\nabla F(\cdot)$	Gradient of function F
∂f	Partial derivative of function f
$O(\cdot)$	Big-O notation, denotes the asymptotic upper bound
$\boldsymbol{u} \prec \boldsymbol{v}$	$\boldsymbol{v}$ strictly dominates $\boldsymbol{u}$
$\boldsymbol{u} \preceq \boldsymbol{v}$	$\boldsymbol{v}$ weakly dominates $\boldsymbol{u}$
$\boldsymbol{u} \parallel \boldsymbol{v}$	$\boldsymbol{u}$ and $\boldsymbol{v}$ are incomparable
$\boldsymbol{u} \not\succ \boldsymbol{v}$	$\boldsymbol{v}$ is non-dominated by $\boldsymbol{u}$

Chapter 1

Introduction

1.1 Overview

The ongoing evolution of 5G networks and the development of 6G systems, which are expected to be commercially available by 2030, are being shaped by the incorporation of artificial intelligence (AI). Consequently, alongside advancements in the physical layer and network architecture, researchers in academia and industry (Ericsson, 2024, Viswanathan and Mogensen, 2020) are actively leveraging AI to optimize the performance and adaptability of existing functionalities and to introduce new capabilities that are unattainable through conventional methods.

Building on this AI-driven transformation, future networks are expected to support *ubiquitous intelligence* (Duan et al., 2023, Cui et al., 2025), enabling a wide range of emerging technologies and accommodating the exponential growth in connected devices, including smartphones, wearable devices, and vehicular sensors. This concept involves the integration of AI across all layers of the network, including the core network, radio access infrastructure, user equipment, and the air interface. Ubiquitous intelligence allows edge devices to become context-aware and to autonomously self-organize, self-heal, self-optimize, and adapt to dynamic environments. Furthermore, it facilitates cooperation among heterogeneous network components, enabling them to collaboratively enhance the overall system performance.

Multi-access edge computing (MEC) is a key technology in 5G and 6G networks that deploys computation and storage resources close to end users, significantly reducing end-to-end latency and mitigating security risks associated with centralized cloud computing. MEC has enabled important applications such as smart cities, vehicular systems, and immersive experiences (Sharma et al., 2024), while meeting the stringent requirements for low latency and high reliability. By incorporating intelligence, edge servers can dynamically adapt to local network conditions to optimize resource allocation and service delivery, and can collaborate to enhance learning. In addition, AI-enabled end devices can autonomously select communication channels and offload tasks, thereby improving their individual performance within the network.

In this thesis, we study three main problems. The first concerns service placement

in MEC networks, where the resource-limited edge servers must decide which services to deploy locally, rather than at the cloud, to maximize the reduction in service delay across users. As both the environment and service demand are dynamic and unknown in advance, decisions must be made under uncertainty. The second problem addresses decentralized task offloading and load balancing in dense MEC networks, where users compete for limited communication and computation resources to complete their tasks within a certain time limit. Solving this requires efficient decentralized strategies that cope with partial knowledge and the randomness of user interactions. The third problem focuses on the optimal deployment of mobile federated learning (FL) clients, specifically unmanned aerial vehicles (UAVs), with the goal of maximizing network coverage and minimizing local update latency. We study this problem under two scenarios: one assuming known sensor distribution, and another considering uncertainty in the activity of energy-harvesting sensors and environmental conditions. When viewed broadly, the first two problems focus on leveraging AI techniques to optimize network performance under uncertainty, while the third highlights how intelligent network design can accelerate and enhance the learning process. All three problems are unified by the need to operate under dynamic and uncertain conditions related to the environment, user behavior, and resource availability.

To address the challenges of MEC under incomplete information, *multi-armed bandit* (MAB) algorithms are employed. In its basic form, a MAB problem is a sequential online learning framework in which a learner selects one option, known as an arm, from a finite set and receives a reward drawn from an unknown probability distribution (Lattimore and Szepesvári, 2020). The classical objective in MABs is to select the arms that minimize the cumulative regret, defined as the difference between the reward accumulated by the learner and that of an oracle that always selects the optimal arm. This setting addresses the exploration–exploitation tradeoff, making it well-suited for scenarios where decisions are made sequentially and evaluated based on immediate rewards as in task offloading and network routing. In contrast, the best arm identification (BAI) setting (Bubeck et al., 2009, Audibert and Bubeck, 2010), a type of pure exploration, aims to identify the arm with the highest expected reward by the end of the learning process. The learner explores the available options and concludes with a single exploitation step. This framework is particularly suited for applications involving long-term decision-making, such as service placement or base station deployment, where the focus is on selecting the most effective configuration rather than maximizing immediate rewards.

In this thesis, we study different types of bandit algorithms based on the problem objectives, the number of objectives or agents, and the structural dependencies among the arms. When contextual information is available, we employ linear bandit models that assume a linear relationship between rewards and feature vectors, thereby enabling more efficient learning. We adopt the BAI setting within linear bandits (Soare, 2015) to address the service placement problem after extending it to a multi-agent case. Additionally, we

model the task offloading problem with many users as a mean-field MAB game (Gummadi et al., 2013), where users act as agents and servers represent arms. In this setting, each user's reward depends on the actions of others, but the large user population allows the bandit dynamics to converge to a mean-field equilibrium. Finally, as the UAV deployment problem is multi-objective, we explore the multi-objective MAB framework with a particular focus on BAI (Drugan and Nowé, 2014). In this problem, the observed reward is a vector, where each component corresponds to one of the objectives, and the goal is to identify the set of Pareto-optimal arms that balance the conflicting objectives.

Although the MAB framework in its different variants is well studied, existing algorithms must be adapted to meet the specific requirements of diverse MEC scenarios. Therefore, this thesis focuses on designing and applying bandit algorithms to address the aforementioned MEC challenges. The main contributions of this thesis are discussed in the next section.

1.2 Main Contributions

The contributions of this thesis include two main parts: (i) modeling key MEC problems using MABs, and (ii) designing and applying MAB algorithms to solve them. In addition, one study addresses FL optimization in UAV-enabled networks using a heuristic approach outside the bandit framework. The main contributions are summarized as follows:

- **Service placement problem:** We consider a small cell network where small base stations (SBSs) are equipped with servers, operating as MEC servers. A service provider can place services either in the network cloud or at the edge. The objective is to identify the service placement that maximizes the reduction in user-perceived service delay compared to cloud deployment. To this end, we leverage contextual information at edge servers, such as long-term average service demand and delay, and formulate the problem as a BAI problem in linear bandits under a fixed-confidence setting, where SBSs act as agents and services correspond to arms. To enable agent collaboration and accelerate learning, we propose a distributed and adaptive multi-agent BAI algorithm for linear bandits. Unlike existing approaches that focus on regret minimization or rely on static allocation strategies, the proposed method adaptively selects actions based on observed data, further reducing the sample complexity per agent for best-arm identification. We also derive theoretical upper bounds on communication overhead and sample complexity, and analyze robustness to communication failures.

- **Task offloading problem:** In this problem, a large number of users aim to offload their tasks to servers capable of completing them within a specific expiry time while achieving a target user distribution across servers. Here, users act as decentralized bandit agents, and servers represent arms. Due to the large user population, we

model the problem as a mean-field MAB game, where each agent's reward depends on both its own server selection and the collective behavior of others. Since servers can have varying costs or accessibility, we propose a reward-scaling mechanism that guides the system toward a target load distribution.

- **Coverage and FL delay in UAV-enabled networks:** We consider a UAV-enabled network in which UAVs collect data from ground sensors and act as FL clients, performing local training on the collected data. We address the joint problem of maximizing coverage and minimizing FL update time under uncertainty in the availability and distribution of energy-harvesting sensors. The problem is formulated as a multi-objective BAI problem under a fixed-budget setting, where the FL server acts as the agent and each joint beamwidth configuration across all UAVs corresponds to an arm. To solve it, we extend the General Sequential Elimination algorithm proposed by Shahrampour et al. (2017) to the multi-objective case and introduce a utility function that maximizes the ratio of expected reward, consisting of both coverage and delay, to the expected energy cost. We derive the statistical distribution of the FL delay and the error probability of the proposed algorithm, and demonstrate its effectiveness in identifying optimal deployment configurations.
- **FL delay minimization in UAV-enabled networks:** We consider a UAV-enabled network where UAVs act as FL clients and address the joint problem of coverage optimization and resource allocation to minimize the FL update time. Unlike prior works that assume data availability at clients, we study the impact of UAV placement on data collection and the FL delay assuming known sensor distribution. To solve this problem, we propose a two-step heuristic: first, we adjust UAV positions to control coverage and the amount of collected data. Then, we apply a fair resource allocation scheme for channel assignment and power control to reduce communication latency. Together, these steps mitigate the straggler effect in FL.

1.3 Thesis Outline

This thesis is organized into three main parts. The first part provides background on MEC and the bandit algorithms used in this work. The second part presents the core scientific contributions. The third part presents the conclusion and discusses potential directions for future work.

Chapter 2 provides background on the motivation for the development and use of MEC. It describes the MEC architecture, main uses, and benefits. It also introduces key decision-making challenges within MEC environments and includes a brief mathematical overview of federated learning, which is later used in Chapters 6 and 7 in the context of edge computing.

Chapter 3 presents an overview of the fundamental bandit algorithms and introduces the different MAB variants employed throughout the thesis. It uses standard bandit terminology to ensure consistency and clarity. For each relevant bandit type, the chapter includes a concise explanation of how it can be applied to model specific MEC problems addressed in the subsequent chapters.

Chapter 4 presents the service placement problem in MEC and the proposed distributed and adaptive BAI algorithm in linear bandits in the fixed confidence setting. This work is based on the following two manuscripts. The journal paper is included in this chapter and extends a previously published conference paper, which is not included in this thesis.

- Mariam Yahya, Aydin Sezgin, and Setareh Maghsudi. Service placement in small cell networks using distributed best arm identification in linear bandits. *IEEE Transactions on Mobile Computing*, pages 1–14, 2026.
- Mariam Yahya, Aydin Sezgin, and Setareh Maghsudi. Service placement using distributed pure exploration in linear bandits. In *33rd European Signal Processing Conference (EUSIPCO)*, pages 2072–2076. IEEE, 2025.

Chapter 5 addresses the problem of decentralized task offloading and load balancing using a mean field MAB game. This chapter corresponds to the following publication:

- Mariam Yahya, Alexander Conzelmann, and Setareh Maghsudi. Decentralized task offloading and load-balancing for mobile edge computing in dense networks. *IEEE Communications Letters*, 28(8):1954–1958, 2024a.

Chapter 6 presents the joint problem of coverage and convergence time optimization in UAV-enabled networks formulated as a multi-objective BAI problem. The following journal paper contains the full version of this work. It extends a previously published conference paper, which is not included in this thesis.

- Mariam Yahya, Setareh Maghsudi, and Slawomir Stanczak. Federated learning in UAV-enhanced networks: Joint coverage and convergence time optimization. *IEEE Transactions on Wireless Communications*, 23(6):6077–6092, 2024b.
- Mariam Yahya, Setareh Maghsudi, and Slawomir Stanczak. Federated learning in UAV-enhanced networks: Joint coverage and convergence time optimization. In *The 27th European Wireless Conference*, pages 1–7, 2022.

Chapter 7 presents the problem of joint coverage optimization and resource allocation in UAV-enabled networks to minimize the FL update time. It presents the following publication:

- Mariam Yahya and Setareh Maghsudi. Joint coverage and resource allocation for federated learning in UAV-enabled networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2476–2481, 2022

Part I

Background

Chapter 2

Multi-Access Edge Computing and Federated Learning

This chapter introduces multi-access edge computing (MEC) and explains the motivation for its development and use in 5G and 6G networks. It then outlines the MEC architecture, key applications, and benefits. The chapter also highlights some of the main decision-making challenges in MEC. Finally, it provides an introduction to federated learning (FL), which is employed as an edge technology in this thesis. Background on the solution framework is presented in the following chapter.

2.1 Multi-Access Edge Computing

2.1.1 Motivation

End devices in 5G and future 6G communication networks support a wide range of applications that demand high computation, energy, and memory resources (Sharma et al., 2024, Malazi et al., 2022). These applications often require low latency, high reliability, and substantial processing capabilities, which exceed the capabilities of resource-constrained end devices. Historically, such computational tasks were offloaded to centralized cloud data centers equipped with large computation and storage resources. However, this approach introduces considerable network latency and potential network congestion due to the long transmission paths between end users and the cloud. Furthermore, the centralized nature of cloud computing raises concerns related to data privacy and security, especially when handling sensitive data. The limitations of cloud computing are further exacerbated in dynamic and resource-constrained environments, such as UAV-enabled networks. In such settings, transmitting over long distances and maintaining connectivity to remote cloud servers is impractical due to limited energy reserves, intermittent backhaul links, and high mobility.

To address these limitations, MEC has emerged as a paradigm that brings computation and storage resources closer to the network edge. By processing tasks closer to end users, MEC reduces end-to-end latency, enables context-aware services, and improves

the overall quality of experience (QoE). Additionally, MEC facilitates collaborative computing and caching strategies among edge servers, thereby improving service reliability and scalability. Originally known as mobile edge computing, the acronym MEC was later redefined by the European Telecommunications Standards Institute - Industry Specification Groups (ETSI ISG) to reflect the support for multiple access technologies, including 4G LTE, 5G, Wi-Fi, and other fixed and wireless connections (Malazi et al., 2022).

2.1.2 MEC Architecture

A typical MEC system is organized into a hierarchical three-tier architecture consisting of the following components:

- **End Devices:** This layer includes user equipment (UE) such as smartphones, wearables, vehicular UE, and IoT devices. These devices are typically responsible for generating data and initiating computation offloading requests and can vary in computational capability, energy availability, and connectivity.

- **Edge Layer:** The intermediate layer consists of MEC servers deployed close to end devices, typically co-located with base stations, Wi-Fi access points, or mounted on mobile platforms such as UAVs. These servers provide localized computing and caching capabilities and may collaborate horizontally to share workloads and balance resource usage.

- **Core Network and Cloud**: The top layer connects edge layer to the centralized cloud via the core network. The cloud layer offers large-scale computation and storage resources. It may be supported by public cloud providers, such as Amazon Web Services (AWS) and Alibaba Cloud, or by private cloud infrastructures.

2.1.3 MEC Applications and Benefits

MEC serves as a foundational enabler for emerging paradigms such as Industry 5.0, which emphasizes human-centric and AI-integrated manufacturing systems (for Research and Innovation, 2024, Sharma et al., 2024). Use cases include predictive maintenance, robotic process automation, and intelligent quality control. In the healthcare sector, MEC supports continuous monitoring through wearables, real-time diagnostic assistance, and remote consultations. In intelligent transportation systems, MEC facilitates V2X (vehicle-to-everything) communication by enabling low-latency processing of sensor data for path planning, collision avoidance, and cooperative driving (Yang et al., 2023). MEC is also instrumental in enabling aerial networks for precision agriculture, environmental monitoring, and automated logistics.

The main benefits of MEC include:

- Reduced latency: By processing or accessing data closer to end users, MEC reduces the communication delay between the user and the edge server.
- Reduced network congestion: By processing or caching services and data at the network edge, MEC reduces core network traffic and mitigates bottlenecks commonly encountered at centralized data centers.
- Enhanced data privacy and security: Processing data locally reduces the amount of data transmitted over the network, thereby mitigating potential privacy risks.
- Distributed and federated learning: MEC enables edge servers to perform distributed learning through techniques such as FL. In FL, MEC nodes act as learning clients that train local models on private data and exchange only model updates with a central aggregator or other clients, thereby preserving data privacy.

2.2 Key Decision Problems in MEC

Many of the key challenges in MEC can be formulated as decision-making problems, especially in environments characterized by limited resources, dynamic conditions, and incomplete information. Depending on the type and availability of information, these challenges can be tackled using optimization techniques or online learning algorithms. The decision-making process must be effective, scalable, and adaptive to ensure the efficient operation of MEC systems. This section outlines several fundamental decision problems commonly encountered in MEC.

2.2.1 Task Offloading and Resource Allocation

Task offloading is the process by which the UE transmits a task, either partially or entirely, for remote processing and receives the result. A key aspect of this process involves deciding whether to offload the task to the cloud or to an edge server and, in some cases, selecting the appropriate edge server. This offloading decision depends on several factors, including the available communication resources and the computational capacity of edge servers.

2.2.1.1 Network Constraints

Wireless communication networks are limited by bandwidth and data rate constraints. UEs transmitting to the same edge server share the uplink channel, reducing the bandwidth per user and increasing transmission delays. Moreover, high user density can increase interference levels, further degrading signal quality and increasing delays.

Accordingly, the uplink data rate between a user k and an edge server m at time t is given by (Proakis and Salehi, 2008)

$$\rho_{k,m,t} = B_{k,m,t} \log_2 \left(1 + \frac{P_{k,m,t} h_{k,m,t}}{\sigma^2_{\mathrm{N},k,m} + I_{k,m,t}} \right), \tag{2.1}$$

where $B_{k,m,t}$ is the bandwidth allocated to user k, $P_{k,m,t}$ is the user's transmission power, $\sigma^2_{\mathrm{N},k,m}$ is the thermal noise power, and $I_{k,m,t}$ is the interference power. Additionally, $h_{k,m,t}$ is the channel gain resulting from both small-scale fading due to multipath propagation and large-scale fading, which includes path loss and shadowing effects caused by obstacles and the distance to the server. The bandwidth allocated to user k is a fraction of the total bandwidth $B_{m,t}$, and it decreases as more users share the same channel.

The time needed to transmit $s_{k,m,t}$ bits from user k to server m at time t is

$$d^{\mathrm{cm}}_{k,m,t} = \frac{s_{k,m,t}}{\rho_{k,m,t}}, \tag{2.2}$$

where $\rho_{k,m,t}$ is given in (2.1). This delay often forms one component of the utility function in decision-making problems where offloading decisions are based on delay considerations, either directly, through the value of the delay, or comparatively, by evaluating differences in delay when offloading to edge server m versus alternative servers.

2.2.1.2 Computational Constraints

The computational resources at the network edge are limited and shared between all users offloading tasks to the server. Offloading users either queue their tasks, resulting in longer waiting times as data is processed sequentially (Choudhury et al., 2024), or share the available computing resources through virtualization techniques (Guo et al., 2024), where each user is allocated a fraction of the total computing capacity. In this work, we consider the latter approach for processing multiple tasks at the server. Specifically, let c be the number of processing cycles required for each bit, and $f_{k,m,t}$ be the CPU computational capacity that edge server m allocates to user k at time t, then the processing (or computation) delay for a user's task of size $s_{k,m,t}$ is

$$d^{\mathrm{proc}}_{k,m,t} = \frac{c s_{k,m,t}}{f_{k,m,t}}. \tag{2.3}$$

If the total CPU capacity of the edge server at time t is $f_{m,t}$, and is shared equally between K users, then $f_{k,m,t} = f_{m,t}/K$. Another challenge in this problem is the uncertainty in the amount of available computational resources at any given time, since the server can be occupied with other tasks.

2.2.1.3 Load Balancing

Load balancing in MEC involves distributing task offloading requests among multiple edge servers to ensure efficient use of available resources and prevent resource over- and under-utilization. Common objectives include minimizing delay and server congestion, promoting fairness in resource allocation, and achieving desired load distributions based on factors such as offloading pricing or energy consumption (Yi et al., 2021).

Traditional static load balancing algorithms often rely on prior knowledge of system parameters and assume fixed user demand. However, in practice, user requests are dynamic and stochastic in both volume and arrival patterns. This makes dynamic algorithms necessary for optimizing performance (Esmaeili et al., 2024).

Load balancing approaches can be broadly categorized into centralized and distributed schemes. In centralized schemes, a global controller collects system information and determines the offloading decisions to optimize a global objective. In contrast, distributed schemes allow users or edge devices to make autonomous decisions based on local observations. In these settings, users may compete for limited edge resources (Liu et al., 2021) to optimize individual performance metrics, such as minimizing task completion delay. Alternatively, they may cooperate (Yi et al., 2021) to jointly optimize a shared objective, such as reducing system-wide latency.

2.2.2 Service Placement

MEC facilitates UEs' access to services by placing some of these services at the network edge, closer to end users. Examples of such services include gaming, health monitoring and intelligence, industrial process management (Sharma et al., 2024) , and video streaming (Khan et al., 2022). However, with the increasing diversity in UE types and the broad range of services that users may request, determining which services to place at the edge has become an increasingly important and complex problem. This complexity arises primarily due to the limited resources available at edge servers. Only a subset of services can be cached and executed locally, while others must be accessed from neighboring edge servers or the cloud, resulting in additional communication delays. The uncertainty in user availability and changing user service demands over time further complicates the problem.

The objectives of service placement include improving QoS metrics, such as user-perceived delay or task completion time. Additional objectives include maximizing the cache hit rate to ensure that more user requests are served by edge servers, and minimizing costs. Costs can arise from renting services from providers, or from operational overheads, such as CPU usage and energy consumption. Another important objective is load balancing to avoid congestion and minimize user delay.

The service selection and placement phase is followed by a resource allocation step, which operates on a much shorter time scale (Zhou et al., 2022). Thus, service placement

decisions must account for the dynamic nature of the environment and aim to optimize performance over time to support long-term deployments. In contrast, resource allocation at the edge must respond to real-time conditions and adapt accordingly. Here, storage resources can often be shared flexibly, whereas CPU and communication resources are typically more limited and must be carefully partitioned to ensure service quality.

It is important to distinguish between problems where service placement decisions are made by edge servers, as discussed above, and those made by users. In the latter setting, a mobile user moves across multiple edge servers and must decide whether to continue accessing the service from the original server, which may increase latency due to data relaying, or to migrate the service to a new edge server. Although migration can reduce latency, it also introduces overhead and potential failure risks (Ouyang et al., 2019). This thesis focuses on the former setting, where decisions are made by the edge servers.

2.3 Federated Learning in MEC

In MEC systems, user equipment and edge servers generate large volumes of data that can be leveraged to train machine learning models for various applications. However, transmitting raw data from distributed devices to centralized servers introduces significant communication overhead and raises privacy concerns. FL addresses these challenges by enabling the devices to collaboratively train machine learning models without sharing raw data. This learning approach aligns well with the MEC architecture, where computation and model training can be distributed across end devices and edge servers.

2.3.1 Overview of Federated Learning and Its Challenges

FL is a distributed machine learning algorithm that enables a set of devices to collaboratively train a shared global model without sharing their raw data. These learning devices, referred to as clients, train their models using their local data and periodically share model updates with a centralized coordinator that aggregates this data and sends an updated model to the clients. In some settings, clients interact directly in a decentralized manner, without relying on a central server. By eliminating the need to transmit raw data, FL preserves user privacy and significantly reduces communication and energy overhead compared to traditional centralized learning approaches. This has motivated the use of FL in many applications such as Google's Gboard for next word prediction (Hard et al., 2018) and in Apple's voice classifier in Siri (Di Sivo et al., 2025).

Despite its advantages, FL introduces several challenges. As model updates are exchanged between clients and the server, communication becomes a major bottleneck due to limited bandwidth, latency, and energy constraints. To address this, many FL algorithms are designed to minimize the number of communication rounds while maintaining model accuracy. This can be achieved through effective resource allocation, client

selection (Zhang et al., 2021), and data compression (Sattler et al., 2020), among others.

Another key issue is the straggler problem, which occurs when one or more clients experience long delays or fail to send their model updates within a certain time window. This delays the aggregation process at the server, which must wait for all local updates, thereby slowing or destabilizing the convergence of the global model. Such delays often stem from heterogeneity in clients' computational or communication resources, differences in clients' data volumes, as well as unstable connections, client mobility, and uncertain availability. In this thesis, we address the straggler issue through careful management of local training data and resource allocation in Chapter 7, and indirectly in Chapter 6 by identifying the network configuration that minimizes delay. Alternative approaches focus on designing FL systems that are robust to partial client participation and asynchronous client updates.

The following sections provide a discussion on FL implementation at the network edge, specifically on UAV-mounted base stations, followed by a formal description of the FL problem.

2.3.2 FL in UAV-Enabled MEC Networks

In MEC systems, FL can be implemented by having edge servers or UE acting as FL clients. In UAV-enabled networks, UAVs operating as base stations can serve as clients that either generate data using onboard sensors or collect data from ground devices and participate in FL. In Chapters 6 and 7, we focus on the latter scenario, where sensors are the primary data sources and UAVs serve as aerial base stations that collect sensor-generated data and participate as clients in the FL process.

In this setting, the FL performance is tightly coupled with UAV placement, which directly impacts the quantity of data collected from sensors. This coupling creates a multi-objective trade-off between communication-based goals, such as maximizing sensor coverage and minimizing latency, and learning-based objectives, such as faster convergence. Consequently, efficient FL in UAV-enabled edge networks requires the joint optimization of UAV deployment and the FL delay. Balancing these often conflicting objectives is critical to overall system performance.

Implementing FL in UAV-enabled networks introduces challenges beyond the classical concerns of fast convergence, low communication overhead, and fairness. These challenges mainly stem from UAV mobility, limited energy and storage capacity, and the dynamic nature of wireless channels. Specifically, air-to-ground channels are subject to time-varying conditions influenced by path loss, fading, and intermittent line-of-sight (LoS) availability. Such impairments can destabilize the learning process, delay model aggregation, and exacerbate the straggler problem.

2.3.3 The Federated Learning Process

Let M denote the number of clients in a FL setting. Each client $m \in [M]$ has a local dataset consisting of D_m samples, denoted as $\{(\boldsymbol{x}_{ml}, y_{ml})\}_{l=1}^{D_m}$, where $\boldsymbol{x}_{ml} \in \mathbb{R}^d$ is the input feature vector and $y_{ml} \in \mathbb{R}$ is the corresponding label. The model is parameterized by $\boldsymbol{\omega} \in \mathbb{R}^d$, and the loss function for each sample $(\boldsymbol{x}_{ml}, y_{ml})$ at client m is $f_m(\boldsymbol{\omega}; \boldsymbol{x}_{ml}, y_{ml})$.

Depending on the learning task, the loss function can take various forms. For example, for ℓ_2-regularized linear regression, the loss function is $f_m(\boldsymbol{\omega}; \boldsymbol{x}_{ml}, y_{ml}) = \frac{1}{2}\big(\langle \boldsymbol{x}_{ml}, \boldsymbol{\omega}\rangle - y_{ml}\big)^2 + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2$, with $y_{ml} \in \mathbb{R}$. Whereas for ℓ_2-regularized logistic regression, the loss function is $f_m(\boldsymbol{\omega}; \boldsymbol{x}_{ml}, y_{ml}) = \log\big(1+\exp\big(-y_{ml}\langle \boldsymbol{x}_{ml}, \boldsymbol{\omega}\rangle\big)\big) + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2$, where $y_{ml} \in \{-1, 1\}$. The properties of the loss function in terms of convexity and smoothness determine the convergence of the FL algorithm and the type of algorithm to be used.

The local loss function of client m over its entire dataset is defined as

$$F_m(\boldsymbol{\omega}) \coloneqq \frac{1}{D_m} \sum_{l=1}^{D_m} f_m(\boldsymbol{\omega}; \boldsymbol{x}_{ml}, y_{ml}). \tag{2.4}$$

The global loss function, representing the weighted aggregation of all local losses, is given by (Dinh et al., 2021)

$$F(\boldsymbol{\omega}) \coloneqq \sum_{m=1}^{M} \frac{D_m}{D} F_m(\boldsymbol{\omega}), \tag{2.5}$$

where $D = \sum_{m=1}^{M} D_m$. The use of D_m/D to weight each client's contribution is standard in algorithms such as the popular federated averaging (FedAvg) algorithm proposed by McMahan et al. (2017). However, alternative FL schemes may employ equal weighting or other aggregation rules depending on design goals, privacy constraints, or network topology.

The objective of FL is to find the model that minimizes the global loss function as given by the following optimization problem:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^d} F(\boldsymbol{\omega}). \tag{2.6}$$

The FL process can be described through the widely used FedAvg algorithm (McMahan et al., 2017). At each global round t, a subset of Q randomly selected clients, where $1 \leq q \leq Q$, participate in local model training. Each participating client q sets its local model to the latest global model, i.e., $\boldsymbol{\omega}_t^q = \boldsymbol{\omega}_t$, and performs E steps of local stochastic gradient descent (SGD) using its local dataset. One step of SGD is given by

$$\boldsymbol{\omega}_{t,e+1}^q \leftarrow \boldsymbol{\omega}_{t,e}^q - \eta \nabla F_q\big(\boldsymbol{\omega}_{t,e}^q\big), \tag{2.7}$$

where $\eta > 0$ is the learning rate, $F_q(\cdot)$ is the local loss function of client q, and $e = 0, \ldots, E-1$ indexes the local training step. Afterwards, each client sends its updated local model $\boldsymbol{\omega}_{t+1}^q$ to the server. The server then aggregates the models using a weighted average based on the size of each client's local dataset:

$$\boldsymbol{\omega}_{t+1} \leftarrow \sum_{m=1}^{M} \frac{D_m}{D} \boldsymbol{\omega}_{t+1}^m. \tag{2.8}$$

The algorithm stops when any of the following conditions is met: a predefined number of rounds is reached, the change in the global model between two consecutive rounds falls below a threshold ϵ, i.e., $\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\| < \epsilon$, or the target learning accuracy is achieved.

While FedAvg is widely used for its simplicity, it has several limitations that affect performance. It performs poorly under non-i.i.d. client data, lacks privacy guarantees, and does not address issues such as high communication overhead, client heterogeneity, and straggler effects. Moreover, its reliance on synchronous updates and stable client availability makes it less suitable for dynamic environments. These shortcomings have motivated the development of more advanced FL algorithms that are robust, communication-efficient, and privacy-preserving. The FEDL algorithm by Dinh et al. (2021), discussed in Chapters 6 and 7, addresses client heterogeneity in data size.

Chapter 3

Bandit Learning Algorithms and Their Applications in Multi-Access Edge Computing

The previous chapter highlighted key challenges in MEC, particularly those arising from uncertainty in the environment, user activity, and complex interactions among network components. Conventional approaches that assume knowledge of some network parameters, such as channel state information or the average number of active users, or rely on estimating them, often fail to capture the real-time dynamics of these networks. To address these limitations, bandit algorithms provide a theoretically grounded framework for sequential decision-making under uncertainty. These algorithms are particularly well-suited for network problems where full system knowledge is unavailable or costly to obtain, and where decisions must adapt to network changes.

This chapter introduces the main concepts and key variants of bandit algorithms that are used in optimizing the performance of MEC in Chapters 4, 5, and 6. The discussion begins with the classical MAB model and progresses to more advanced settings, including linear bandits, pure exploration, multi-objective bandits, and mean-field bandit games.

3.1 Introduction

A bandit problem is a sequential decision-making process in which a learner, also called an agent, interacts with an unknown environment. In its seminal form, a bandit problem is played over n rounds and in each round $t \in [n]$, the learner selects an action, also called an arm, a_t from the set of available actions $\mathcal{A}$. It then receives a stochastic reward $r_t \in \mathbb{R}$, sampled from an unknown probability distribution associated with the chosen action. The learner's objective may include, among others, maximizing the cumulative reward or identifying the arm with the highest expected reward under certain constraints. In either case, the challenge lies in making decisions under uncertainty, as the learner has no prior knowledge of the actions' reward distributions.

At each round, the learner selects an action based on its *history* up to time t, defined as

the sequence $H_{t-1} = (a_1, r_1, \ldots, a_{t-1}, r_{t-1})$, which considers all previously chosen actions and their observed rewards. The action selection is governed by the *policy*, a rule that maps the history to a probability distribution over the action set $\mathcal{A}$. The environment is specified by a collection of reward distributions $\boldsymbol{\nu} = (\nu_a : a \in \mathcal{A})$, where each ν_a is a probability distribution over the real numbers $\mathbb{R}$. The reward observed at round $t \in [n]$ by the learner upon selecting $a_t \in \mathcal{A}$ follows $r_t \sim \nu_{a_t}$. These distributions are assumed to be stationary and independent across rounds, conditioned on the actions.

For each action $a \in \mathcal{A}$, the expected reward is given by $\mu_a(\boldsymbol{\nu}) = \int_{-\infty}^{\infty} x d\nu_a(x)$, which is assumed to be finite. The optimal action is defined as the one that maximizes the expected reward and is given by

$$\mu^*(\boldsymbol{\nu}) \coloneqq \max_{a \in \mathcal{A}} \mu_a(\boldsymbol{\nu}). \tag{3.1}$$

The performance of bandit algorithms is typically evaluated in terms of regret, which measures the loss incurred by the learner relative to the optimal policy.

3.1.1 Types of Bandit Problems

Different types of bandit problems arise depending on factors such as the learning objective, the number of objectives and agents, the existence of a structural relationship between actions, and the use of contextual information. These types are summarized in Fig. 3.1. The main categories can be outlined as follows:

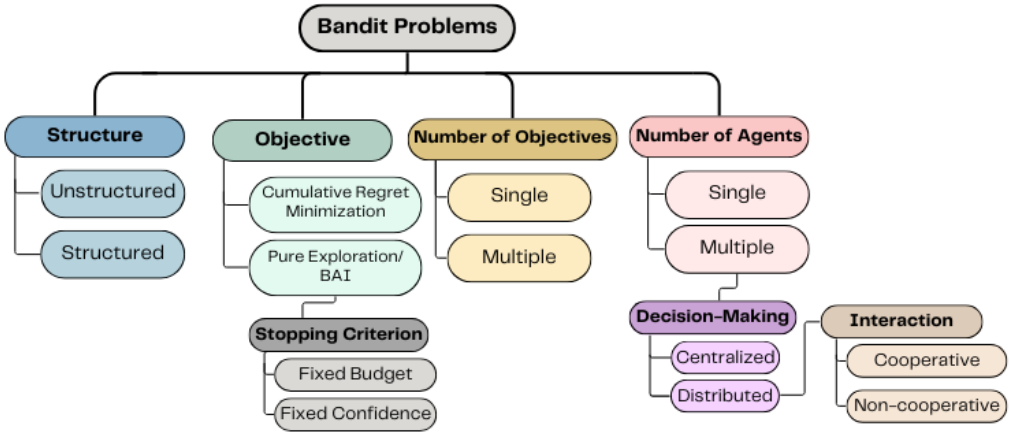


Figure 3.1: Types of bandit problems.

- **Objective**
  - **Cumulative regret minimization**: The goal is to maximize the expected total reward over the time horizon, which is equivalent to minimizing the cumulative regret relative to the optimal action.

  - **Pure exploration**: The goal is to identify the optimal action, typically the one with the highest expected reward, or the one that satisfies a predefined threshold or constraint. In this setting, the exploration phase is separate from exploitation, and performance is measured by the probability of correctly identifying the optimal action. This thesis focuses on the best arm identification (BAI) problem under both the fixed budget (number of sampling rounds) and the fixed confidence (probability of correct identification) settings.

- **Structure**
  - **Unstructured bandits**: The reward of each action is independent of the others. Learning must occur separately for each arm.
  - **Structured bandits**: There exists some structure between the rewards of different actions, meaning that selecting one action can reveal some information about others. Exploiting such structure allows for more efficient learning.

- **Number of Objectives**
  - **Single-objective**: The learner receives a scalar reward for each selected action and aims to find a single optimal solution.
  - **Multi-objective**: The leaner receives a reward vector for each selected action. Each element of the vector corresponds to a different, often conflicting, objective. Trade-offs between objectives typically exist, resulting in multiple optimal solutions.

- **Number of agents**
  - **Single-agent**: A single agent interacts with the environment to achieve one of the objectives mentioned above.
  - **Multi-agent**: Multiple agents interact within a shared or separate environment. These agents may cooperate to achieve a common goal or act competitively with individual goals. The presence of multiple agents can cause non-stationarity, as the reward distribution of each agent may be influenced by the actions of others. Decision-making can be centralized or distributed.

3.1.2 Modeling Communication Problems as Bandit Problems

Depending on the application, network components such as user equipment, base stations, or edge servers may function as decision-making agents. The action set includes the options available to an agent, such as possible base station locations, services for deployment in MEC, or servers for task offloading. Each action yields a stochastic reward that captures the uncertainty in the network and guides future decisions. Because the bandit

model depends on the specific problem context, detailed modeling for the key problems addressed in this thesis is provided within their respective chapters.

3.2 The Classical Bandit Setting

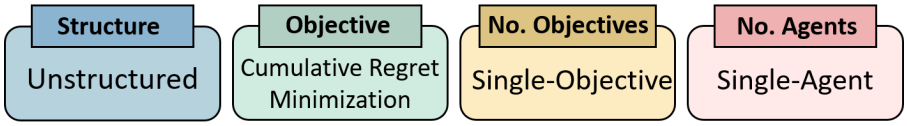


Figure 3.2: The classical bandit setting.

A widely studied setting in the MAB framework involves minimizing the expected cumulative regret in an unstructured bandit problem over a time horizon n. This setting is common in applications such as online routing, where the learner must balance exploring new actions with exploiting those known to give high rewards, a trade-off known as the exploration-exploitation dilemma. In each round $t \in [n]$, the agent selects an arm $a_t \in [K]$, where $[K]$ denotes the fixed set of arms. Then, the environment samples a reward $r_t \in \mathbb{R}$ from a distribution ν_{a_t}, unknown to the agent. The agent's objective is to maximize the expected total reward, $G(n) = \mathbb{E}_\pi \left[\sum_{t=1}^{n} r_t\right]$.

To provide a standardized measure of performance across different bandit problems, the problem is often reformulated as minimizing the expected cumulative regret. Given a bandit instance $\boldsymbol{\nu}$, the expected cumulative regret for a policy π is defined as

$$R_n(\pi, \boldsymbol{\nu}) = \mathbb{E}_\pi \left[n\mu^*(\boldsymbol{\nu}) - \sum_{t=1}^{n} \mu_{a_t}(\boldsymbol{\nu}) \right], \tag{3.2}$$

where $\mu^*(\boldsymbol{\nu}) = \max_{k\in[K]} \mu_k(\boldsymbol{\nu})$ is the expected reward of the best arm, and $\mu_{a_t}(\boldsymbol{\nu})$ is the expected reward of the arm selected at time t. By definition $R_n(\pi, \boldsymbol{\nu}) \geq 0$. The goal is to follow a policy that achieves *sublinear regret* for all bandit instances $\boldsymbol{\nu}$ such that

$$\lim_{n\to\infty} \frac{R_n(\pi, \boldsymbol{\nu})}{n} = 0, \tag{3.3}$$

ensuring that the average regret vanishes as n grows. We omit the dependence on π and $\boldsymbol{\nu}$ since they are clear from the context and write the regret as $R_n = \mathbb{E}_\pi \left[n\mu^* - \sum_{t=1}^{n} \mu_{a_t}\right]$.

The Upper Confidence Bound Algorithm

Several algorithms have been developed to minimize the cumulative regret. A foundational and widely used one is the upper confidence bound (UCB) algorithm (Auer et al.,

2002), based on the principle of *optimism in the face of uncertainty*. According to this principle, at each round, the agent selects the arm with the highest upper confidence bound, which is the sum of the empirical mean reward and a confidence bonus that accounts for uncertainty due to limited sampling. This approach balances the exploitation of arms showing high rewards with the exploration of less sampled arms.

To describe UCB formally, we consider a bandit setting where the arms' reward distributions are assumed to be sub-Gaussian, a general class that includes Bernoulli and Gaussian distributions and allows for tight concentration inequalities. A random variable X is σ-sub-Gaussian if for $\lambda \in \mathbb{R}$, it holds that $\mathbb{E}[\exp(\lambda X)] \leq \exp(\lambda^2\sigma^2/2)$ (Lattimore and Szepesvári, 2020, Definition 5.2).

Let $r_1, r_2, \ldots, r_\ell$ be a sequence of independent and identically distributed (i.i.d.) sub-Gaussian random variables corresponding to the rewards observed from pulling an arm. The *empirical mean* of these rewards is $\hat{\mu}_\ell = \frac{1}{\ell}\sum_{t=1}^{\ell} r_t$, which concentrates around the true mean μ with high probability. Under the sub-Gaussian assumption on rewards, the following concentration inequality holds for $\delta \in (0,1)$:

$$\mathbb{P}\left(\mu \geq \hat{\mu}_\ell + \sqrt{\frac{2\sigma^2 \log(1/\delta)}{\ell}}\right) \leq \delta. \tag{3.4}$$

This inequality provides the basis for constructing high-probability upper confidence bounds on the arm's mean reward.

After an initial phase in which each arm is played once, the UCB algorithm selects at each time the arm with the highest UCB index, defined as follows. At time t, let $T_k(t-1)$ be the number of times arm k has been played up to time $t-1$, and let $\hat{\mu}_k(t-1)$ be its empirical mean. The UCB index for arm k is defined as

$$\mathrm{UCB}_k(t-1) = \underbrace{\hat{\mu}_k(t-1)}_{\text{exploitation}} + \underbrace{\sqrt{\frac{2\sigma^2 \log(1/\delta)}{T_k(t-1)}}}_{\text{exploration}}. \tag{3.5}$$

The exploitation term favors selecting arms with high observed rewards, while the exploration term favors sampling underexplored arms with high uncertainty. At time t, the selected arm is $a_t = \arg\max_{k\in[K]} \mathrm{UCB}_k(t-1)$. This policy ensures that all arms are eventually explored, but increasingly samples most promising ones as more data is gathered.

3.3 Linear Bandits

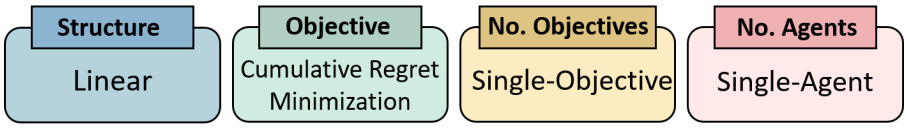


Figure 3.3: The linear bandit setting.

In many scenarios, the bandit problem has an underlying structure that allows the learner to gain information about other arms when one arm is pulled. Learning this structure can significantly reduce the complexity of the bandit problem, especially when the number of arms exceeds the learning horizon n, making the exploration of individual arms infeasible. A common example appears in movie recommendation systems, where the rewards, and thus the arm selections, are influenced by user and movie features. Similarly, in MEC, using information such as service characteristics or user locations can improve decision-making.

In each round t of the linear bandit problem, the learner selects an arm a_t from the set of available arms $\mathcal{A}_t$, associated with an action vector $\boldsymbol{x}_{a_t} \in \mathbb{R}^d$. The learner then observes a reward r_t that is a noisy linear function of the chosen action. Formally,

$$r_t = \langle \boldsymbol{x}_{a_t}, \boldsymbol{\theta}^* \rangle + \eta_t, \tag{3.6}$$

where $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is an unknown parameter vector defining the linear relation between actions and expected rewards, and η_t is a zero-mean σ-sub-Gaussian noise term.

The linear bandit framework includes several important special cases depending on the set of action vectors $\mathcal{X}_t \subset \mathbb{R}^d$. In the case of *contextual linear bandits*, assuming that there is a fixed set of K arms, each round t the learner observes a context $\boldsymbol{\xi}_t$ belonging to the context space Ξ. A known feature mapping $\psi : \Xi \times [K] \to \mathbb{R}^d$ is applied to the context-arm pairs to generate the action set $\mathcal{X}_t = \{\psi(\boldsymbol{\xi}_t, k) : k \in [K]\}$. By leveraging this contextual information, the learner can make more informed decisions. In Chapter 4, we consider a simplified setting where each arm $k \in [K]$ is associated with a fixed context vector $\boldsymbol{\xi}_k \in \mathbb{R}^d$, which also serves as its action vector. In this case, the action set is time-invariant and is given by $\mathcal{X} = \{\boldsymbol{x}_k = \boldsymbol{\xi}_k : k \in [K]\}$. Another special case of linear bandits is the classical MAB problem, discussed in Section 3.2. It is the case when the action set is the canonical basis vectors in $\mathbb{R}^d$, i.e., $\mathcal{X} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K\}$, where $\boldsymbol{e}_k$ denotes the k-th standard basis vector.

The learner's objective in this linear bandit setting is to minimize the expected cumulative regret. Let the optimal arm at time t be defined as $a_t^* := \arg\max_{a \in \mathcal{A}_t} \boldsymbol{x}_a^\top \boldsymbol{\theta}^*$. Then, the

expected cumulative regret is given by

$$R_n = \mathbb{E}_\pi \left[\sum_{t=1}^{n} \left(\boldsymbol{x}_{a_t^*} - \boldsymbol{x}_{a_t} \right)^\top \boldsymbol{\theta}^* \right]. \tag{3.7}$$

Since the true parameter vector $\boldsymbol{\theta}^*$ is unknown, the learner constructs an estimate based on the selected actions and observed rewards in previous rounds.

The OFUL Algorithm

A widely used approach for regret minimization in linear bandits is the *Optimism in the Face of Uncertainty for Linear bandits (OFUL)* (Abbasi-Yadkori et al., 2011), which generalizes the classical UCB algorithm. In each round t, the algorithm finds the regularized least square estimate (LSE) of $\boldsymbol{\theta}^*$ based on the history of observed actions and rewards. For regularization parameter $\lambda \geq 0$, the estimate is the minimizer of the loss function and is given by

$$\hat{\boldsymbol{\theta}}_t = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\arg\min} \left(\sum_{s=1}^{t} (r_s - \langle \boldsymbol{\theta}, \boldsymbol{x}_{a_s} \rangle)^2 + \lambda \|\boldsymbol{\theta}\|^2 \right), \tag{3.8}$$

choosing $\lambda > 0$ guarantees that the loss function has a unique minimizer as the actions $\boldsymbol{x}_{a_1}, \ldots, \boldsymbol{x}_{a_t}$ do not always span $\mathbb{R}^d$. The solution to (3.8) is

$$\hat{\boldsymbol{\theta}}_t = \boldsymbol{V}_t^{-1} \sum_{s=1}^{t} \boldsymbol{x}_{a_s} r_s, \tag{3.9}$$

where $\boldsymbol{V}_t = \lambda \boldsymbol{I} + \sum_{s=1}^{t} \boldsymbol{x}_{a_s} \boldsymbol{x}_{a_s}^\top$ is the $d \times d$ design matrix.

Confidence Ellipsoid

The confidence set for the unknown $\boldsymbol{\theta}^*$ is an ellipsoid centered at the current estimate $\hat{\boldsymbol{\theta}}_t$, the principal axis is the eigenvectors of $\boldsymbol{V}_t$ and the corresponding length is the reciprocal of the square roots of the eigenvalues. As t increases, the eigenvalues typically increase, and the volume of the ellipse shrinks, reflecting increasing confidence in the estimate $\hat{\boldsymbol{\theta}}_t$.

Let $\{\eta_t\}_{t=1}^{\infty}$ be conditionally σ-sub-Gaussian noise with $\sigma \geq 0$, and assume that the parameter vector satisfies $\|\boldsymbol{\theta}^*\| \leq S$. Then, for any $\delta \in (0,1)$, it holds with probability at least $1-\delta$ that $\boldsymbol{\theta}^*$ lies in the confidence set (or ellipsoid) given by (Abbasi-Yadkori et al., 2011, Theorem 2)

$$\mathcal{C}_t = \left\{ \boldsymbol{\theta} \in \mathbb{R}^d : \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t\|_{\boldsymbol{V}_t} \leq \sigma \sqrt{2 \log \frac{\det(\boldsymbol{V}_t)^{1/2} \det(\lambda \boldsymbol{I})^{-1/2}}{\delta}} + \lambda^{1/2} S \right\}. \tag{3.10}$$

Arm Selection and Regret

The UCB index for arm $a \in \mathcal{A}_t$ in the OFUL algorithm is $\mathrm{UCB}_a(t) = \max_{\boldsymbol{\theta} \in \mathcal{C}_t} \langle \boldsymbol{\theta}, \boldsymbol{x}_a \rangle$, and the selected arm at round t is

$$a_t = \arg\max_{a \in \mathcal{A}_t} \mathrm{UCB}_a(t). \tag{3.11}$$

Let $\lambda \geq 1$ and suppose that for all $a \in \mathcal{A}_t$, $\langle \boldsymbol{x}_a, \boldsymbol{\theta}^* \rangle \in [-1, 1]$, and $\|\boldsymbol{x}_a\| \leq L$. Then, with probability at least $1 - \delta$, the regret of the OFUL algorithm after n rounds satisfies (Abbasi-Yadkori et al., 2011, Theorem 3)

$$R_n \leq 4\sqrt{nd \log\left(\lambda + \frac{nL}{d}\right)} \left(\lambda^{1/2} S + \sigma \sqrt{2\log(1/\delta) + d\log\left(1 + \frac{nL}{\lambda d}\right)}\right). \tag{3.12}$$

3.4 Pure Exploration

In Sections 3.2 and 3.3, exploration and exploitation occurred simultaneously within each round, which is suitable for settings where each action yields an immediate cost or reward, and the goal is to minimize cumulative regret, as in medical treatments or network routing. In contrast, pure exploration separates exploration from exploitation. In the case of best arm identification, the objective is not to optimize performance during the learning phase but rather to identify the best arm, defined as the one with the highest expected reward, by the end of the exploration phase. This setting is used in applications where actions can be tested in advance, and the final selection is used repeatedly. Examples include evaluating cosmetic products, optimizing webpage layouts, or selecting the most reliable communication channel for data transmission (Audibert and Bubeck, 2010).

Best arm identification can take two settings:

- **The fixed budget setting:** the learner aims to find the best arm within a fixed number of rounds (budget) n, while minimizing the probability of error.

- **The fixed confidence setting:** the learner aims to identify the best arm with confidence at least $1 - \delta$ in the minimum number of rounds.

The BAI strategies differ between structured and unstructured bandits. This thesis focuses on the fixed-budget setting for unstructured bandits and the fixed-confidence setting for linear bandits.

3.4.1 BAI in the Fixed Budget Setting

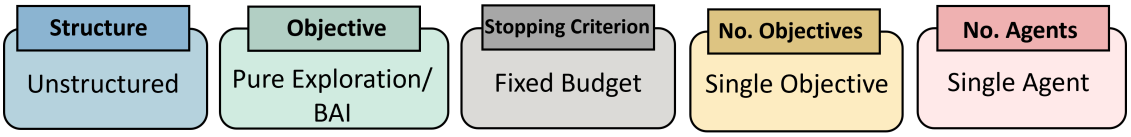


Figure 3.4: BAI in the fixed budget setting.

In this setting, there is a set of K arms, where each arm $k \in [K]$ is associated with an unknown reward distribution ν_k. The learner has a budget of n sampling rounds, and must recommend an arm J_n as the best at the end of the exploration phase. It is assumed that there exists a unique optimal arm k^* such that $\mu_{k^*} = \mu^*$, and that all rewards are bounded in the range $[0,1]$. The quality of the recommendation is evaluated using the error probability, which is defined as the probability of recommending a suboptimal arm $k \neq k^*$:

$$\Pr(n) = \mathbb{P}(J_n \neq k^*) = \sum_{k \neq k^*} \mathbb{P}(J_n = k). \tag{3.13}$$

Since regret is commonly used in the bandit literature, it is helpful to clarify its connection to the error probability in BAI. The *simple regret* is defined as the loss in expected reward resulting from choosing arm J_n instead of the optimal arm, formally,

$$R_n^{\text{Simple}} = \mu^* - \mu_{J_n}. \tag{3.14}$$

For any suboptimal arms $k \neq k^*$ the *suboptimality gap* is defined as $\Delta_k := \mu^* - \mu_k$. The relation between the error probability and the expected simple regret is given as (Audibert and Bubeck, 2010)

$$\mathbb{E}_\pi \left[R_n^{\text{Simple}} \right] = \sum_{k \neq k^*} \mathbb{P}(J_n = k) \Delta_k \leq \Pr(n). \tag{3.15}$$

The error probability and the expected simple regret behave similarly up to a second-order term.

As the sampling budget is limited to n rounds, the objective is to find a sample allocation strategy that minimizes the error probability. A common solution approach is based on *sequential elimination*, where all arms are initially considered potentially optimal. Then, as the algorithm progresses, arms that are identified as suboptimal based on their expected rewards are eliminated. The algorithm runs until only one arm remains, which is then recommended as the best. The main challenge lies in ensuring that eliminations are made with high confidence since eliminated arms are not restored.

Elimination-based algorithms differ in how they allocate the sampling budget and in the number of arms eliminated in each phase. To compare different strategies and

express the difficulty in distinguishing the optimal arm from suboptimal ones, the notion of problem *hardness* is used. It is defined as $H_1 := \sum_{k=1}^{K} \frac{1}{\Delta_k^2}$. It is important to note that no single algorithm consistently outperforms others as their performance depends on the problem instance, specifically on the suboptimality gaps $\Delta_k, k \neq k^*$.

3.4.1.1 Algorithms for BAI in the Fixed Budget Setting:

Successive Rejects (Succ-Rej) (Audibert and Bubeck, 2010): This algorithm divides the total budget n into phases. In each phase $\phi \in \{1, \ldots, K-1\}$, all remaining arms are equally pulled for $n_\phi - n_{\phi-1}$ times, where $n_\phi = \left\lceil \frac{1}{\overline{\log}(K)\frac{n-K}{K+1-\phi}} \right\rceil$ rounds and $\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$. This phase length ensures that Succ-Rej achieves the optimal convergence rate up to a logarithmic factor (Audibert and Bubeck, 2010). At the end of each phase, the arm with the lowest estimated mean reward is eliminated. The last remaining arm is identified as the best.

The upper bound on $\Pr(n)$ is given in Table 3.1, with $H_2 = \max_{i \in [K]} i\Delta_{(i)}^{-2}$, where (i) denotes the i-th best arm. It follows from a union bound over all phases and arms, requiring high confidence estimates of the arms in every phase to avoid mistakenly eliminating the best arm. However, this can lead to excessive sampling of clearly suboptimal arms, an issue that be addressed by eliminating a constant fraction of the arms in each round.

Sequential Halving (Seq-Halv) (Karnin et al., 2013): This algorithm eliminates half of the remaining arms in each phase, resulting in $\lceil \log_2 K \rceil$ phases. In phase ϕ, each remaining arm is sampled $n_\phi = \left\lfloor \frac{n}{|K_\phi| \lceil \log_2 K \rceil} \right\rfloor$, where $|K_\phi|$ denotes the number of arms in phase ϕ. The bound on the error probability is given in Table 3.1. Unlike Succ-Rej, which requires high-confidence estimates in every phase, Seq-Halv progressively allocates more samples to the most promising arms as the algorithm advances.

General Sequential Elimination (Shahrampour et al., 2017): This algorithm generalizes both Succ-Rej and Seq-Halv, and extends classical sequential elimination algorithms by dividing the budget remaining at the end of each phase according to a non-linear function of the number of remaining arms. One special case is the nonlinear sequential elimination (N-Seq-El) algorithm, where one arm is eliminated in each phase and the remaining budget is divided non-linearly among the remaining arms.

Algorithm 3.1 presents the general sequential elimination algorithm. In each phase, $\boldsymbol{B}_\phi$ is the set of eliminated arms and b_ϕ is the number of eliminated arms. Also, $\boldsymbol{G}_\phi$ is the set of arms remaining at the start of phase ϕ, with size $g_\phi = \sum_{i=\phi}^{\Phi} b_i + 1$. The sequence $\{z_\phi\}_{\phi=1}^{\Phi}$ is a positive and increasing sequence that controls the budget allocation. For the N-seq-El it is $z_\phi = (K - \phi + 1)^p$, with $p > 0$, and for Succ-Rej, $p = 1$.

Algorithm 3.1 The General Sequential Elimination Algorithm (Shahrampour et al., 2017)

1: **Input:** budget n, sequence $\{z_\phi, b_\phi\}_{\phi=1}^{\Phi}$.
2: **Initialize:** $\boldsymbol{G}_1 = [K], n_0 = 0$
3: **Output:** $\boldsymbol{G}_{\Phi+1}$
4: Let

$$C = \frac{1}{z_\Phi} + \sum_{r=1}^{\Phi} \frac{b_\phi}{z_\phi}, \qquad n_\phi = \left\lceil \frac{n-K}{C z_\phi} \right\rceil, \quad \forall \phi \in [\Phi] \tag{3.16}$$

5: **for** $\phi = 1, \ldots, \Phi$: **do**
6: Sample each arm $\boldsymbol{G}_\phi$ times for $n_\phi - n_{\phi-1}$ times.
7: Discard the worst b_ϕ arm, with $\boldsymbol{B}_\phi$ being the set of b_ϕ arms with the smallest average rewards.
8: Let $\boldsymbol{G}_{\phi+1} = \boldsymbol{G}_\phi \backslash \boldsymbol{B}_\phi$.
9: **end for**

Table 3.1 gives the number of phases Φ, elimination strategy, and the upper bound on the error probability for the three algorithms described above.

Algorithm	Φ	Eliminate	Upper bound on $\Pr(n)$
Succ-Rej	$K-1$	$b_\phi = 1$	$\frac{K(K-1)}{2} \exp\left(-\frac{n-K}{\log(K) H_2}\right)$
Seq-Halv	$\lceil \log_2 K \rceil$	$g_{\phi+1} = \lceil g_\phi / 2 \rceil$	$3 \log_2 K \exp\left(-\frac{n}{8 H_2 \log_2 K}\right)$
N-Seq-El	$K-1$	$b_\phi = 1$	$\Phi \max_{\phi \in [\Phi]} \{b_\phi\} \exp\left(-\frac{n-K}{C} \min_{\phi \in [\Phi]} \left\{ \frac{2\Delta^2_{g_{\phi+1}+1}}{z_\phi} \right\}\right)$

Table 3.1: Characteristics and performance of sequential elimination algorithms.

In Chapter 6, we extend the General Sequential Elimination algorithm to a multi-objective, cost-aware setting, where the agent observes a reward vector whose dimension corresponds to the number of objectives and uses a utility function such that the optimal arm is the one that maximizes the ratio of expected reward to expected energy cost.

3.4.2 BAI for Base Station Deployment

In Chapter 6, we use the fixed budget setting to determine the optimal placement of mobile base stations in a random environment with unknown user distribution and channel conditions. In our work, the base stations are UAVs that also operate as FL clients. Instead of relocating the UAVs, coverage is adjusted by controlling the UAVs' beamwidths. Each candidate bandwidth configuration is modeled as an arm with rewards reflecting performance metrics such as computation delay, user coverage, or energy consumption. Due to UAVs' energy constraints, the exploration budget is limited by the energy allocated for

exploration. To handle the multi-objective nature of the problem, we extend the General Sequential Elimination algorithm to accommodate multiple rewards and propose a new utility function that considers the energy cost of each arm selection.

3.4.3 BAI in the Fixed Confidence Setting

The objective in the fixed confidence setting is to identify the optimal arm with a predefined confidence in the minimum number of samples. Accordingly, the performance of BAI strategies is mainly determined by the sample complexity. Let $\mathcal{F}_t = \sigma(a_1, r_1, \ldots, a_t, r_t)$ be the sigma-algebra generated by the observations up to time t. Any BAI strategy in the fixed confidence setting is characterized by three main components:

- Sampling rule $\{a_t\}_t$, where a_t is $\mathcal{F}_{t-1}$ measurable.
- Stopping rule τ, which is the stopping time with respect to $\mathcal{F}_t$.
- Decision rule $\hat{a}_\tau$, which is an $\mathcal{F}_\tau$-measurable decision of the best arm at time τ.

3.4.3.1 BAI in Linear Bandits

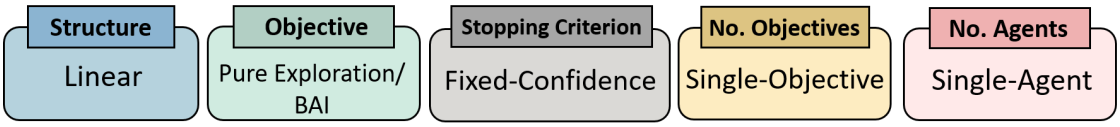


Figure 3.5: BAI in linear bandits in the fixed-confidence setting.

As we will explain shortly, identifying the best arm requires sequentially selecting arms and estimating $\boldsymbol{\theta}^*$ using, for example, the LSE. Since this estimate and its confidence bound depend on the allocation $\mathbf{x}_t = \{\boldsymbol{x}_{a_1}, \boldsymbol{x}_{a_2}, \ldots, \boldsymbol{x}_{a_t}\}$, it is essential to distinguish between static and adaptive allocation strategies.

In *static allocation*, the sequence of arm pulls is fixed in advance and does not depend on the observed rewards. For such a fixed sequence $\mathbf{x}_t$, and assuming the noise η_t is bounded in $[-B_\eta, B_\eta]$, the estimate $\boldsymbol{\theta}^*$ satisfies the following inequality with probability at least $1-\delta$ (Xu et al., 2018):

$$|\boldsymbol{x}^\top \boldsymbol{\theta}^* - \boldsymbol{x}^\top \hat{\boldsymbol{\theta}}_t| \leq 2B_\eta \|\boldsymbol{x}\|_{\boldsymbol{V}_{\mathbf{x}_t}^{-1}} \sqrt{\log\left(6t^2K/(\pi^2\delta)\right)}, \tag{3.17}$$

where $\boldsymbol{V}_{\mathbf{x}_t} = \sum_{s=1}^{t} \boldsymbol{x}_{a_s} \boldsymbol{x}_{a_s}^\top$ is the design matrix resulting from the sequence $\mathbf{x}_t$, K is the number of arms.

In contrast, *adaptive allocation* selects each arm a_t based on the rewards observed in previous rounds. For such an adaptive sequence $\mathbf{x}_t$, the following inequality holds, with

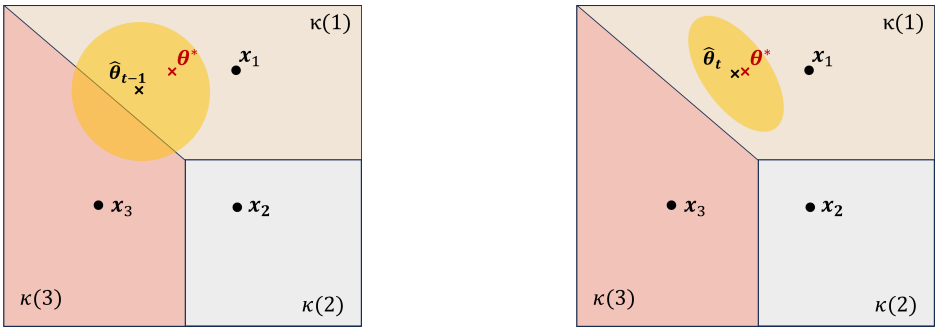
probability at least $1-\delta$ (Abbasi-Yadkori et al., 2011, Theorem 2):

$$|\boldsymbol{x}^\top \boldsymbol{\theta}^* - \boldsymbol{x}^\top \hat{\boldsymbol{\theta}}_t| \leq \|\boldsymbol{x}\|_{(\boldsymbol{V}^\lambda_{\mathbf{x}_t})^{-1}} \left(\sigma \sqrt{d \log\left(\frac{\lambda + tL^2}{\lambda \delta} \right)} + \lambda^{1/2} \|\boldsymbol{\theta}^*\| \right), \tag{3.18}$$

where $\boldsymbol{V}^\lambda_{\mathbf{x}_t} = \lambda \boldsymbol{I} + \sum_{s=1}^t \boldsymbol{x}_{a_s} \boldsymbol{x}_{a_s}^\top$, and $\|\boldsymbol{x}\| \leq L$. The $\sqrt{d}$ factor in (3.18) reflects the additional uncertainty due to the adaptive nature of the sampling strategy.

Geometrically, identifying the best arm in linear bandits can be viewed as partitioning the d-dimensional space into K cones, as illustrated in Fig. 3.6 for a two-dimensional case with three arms. Each cone corresponds to the set of parameter vectors $\boldsymbol{\theta}$ for which a specific arm is optimal. Mathematically, the cone associated with arm $k \in [K]$ is defined as $\kappa(k) = \left\{ \boldsymbol{\theta} \in \mathbb{R}^d : \ k = \arg\max_{a \in [K]} \boldsymbol{x}_a^\top \boldsymbol{\theta} \right\}$.

Identifying the best arm requires estimating the unknown parameter vector $\boldsymbol{\theta}^*$. This often necessitates pulling suboptimal arms to refine the estimates of the parameter vector and reduce the uncertainty about the optimal arm. In each round, the learner selects an arm and updates the confidence set $\mathcal{C}_t$ centered at the current estimate $\hat{\boldsymbol{\theta}}_t$, illustrated in yellow in Fig. 3.6. The optimal arm is identified once the confidence set is entirely contained within a single cone, and the corresponding arm is returned as the best, as in Fig. 3.6b where arm 1 is optimal. If the confidence set overlaps multiple cones, as in Fig. 3.6a, ambiguity remains about which arm is optimal.



(a) The confidence set at time $t-1$. Since it intersects both cones $\kappa(1)$, and $\kappa(3)$, arms 1 and 3 remain potential candidates for the optimal arm.

(b) The confidence set at time t. Arm 1 is identified as optimal because the confidence set lies entirely within cone $\kappa(1)$.

Figure 3.6: Illustration of BAI in linear bandits. Each colored region represents the cone in parameter space where a specific arm $k \in \{1,2,3\}$ has the highest expected reward.

The objective in the fixed confidence setting is to minimize the expected number of samples, $\mathbb{E}[\tau]$, required to identify the best arm with probability at least $1-\delta$. A lower bound on $\mathbb{E}[\tau]$ can be derived using the change of distribution method, which compares two bandit problems with identical arm sets but different parameter vectors. It shows that for a δ-PAC (Probably Approximately Correct) static allocation strategy we have (Soare,

2015, Theorem 3.1):

$$\mathbb{E}[\tau] \geq 2\log\left(\frac{1}{2\delta}\right) \max_{\boldsymbol{y}\in\mathcal{Y}^*} \frac{\|\boldsymbol{y}\|^2_{\boldsymbol{V}_{\boldsymbol{\omega}}^{-1}}}{\Delta(\boldsymbol{y})^2}, \tag{3.19}$$

where $\mathcal{Y}^* = \{\boldsymbol{y} = \boldsymbol{x}_{a^*} - \boldsymbol{x}_a, a \in [K]\}$ is the set of differences between the best arm and all suboptimal arms, $\Delta(\boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{\theta}^*$ is the reward gap along direction $\boldsymbol{y}$, $\boldsymbol{\omega}$ is a probability vector representing the allocation probability, satisfying $\sum_{a\in[K]} \boldsymbol{\omega}(\boldsymbol{x}_a) = 1$, and $\boldsymbol{V}_{\boldsymbol{\omega}} = \sum_{a\in[K]} \boldsymbol{\omega}(\boldsymbol{x}_a)\boldsymbol{x}_a\boldsymbol{x}_a^\top$ is the corresponding design matrix.

This lower bound characterizes the fundamental limit on sample complexity and serves as the foundation for designing sample allocation strategies.

3.4.3.2 Algorithms for BAI in Linear Bandits

The foundational work by Soare et al. (2014) introduced several algorithms for BAI in linear bandits. The first, called the $\mathcal{XY}$-Oracle, assumes that the oracle has prior knowledge of the true parameter vector $\boldsymbol{\theta}^*$, which allows it to know the optimal cone. While this assumption is unrealistic, this strategy establishes a lower bound on the sample complexity. The arm selection rule is $\mathbf{x}_t^* = \arg\min_{\mathbf{x}_t} \max_{y\in\mathcal{Y}^*} \|y\|_{\boldsymbol{V}_{\mathbf{x}_t}^{-1}} / \Delta(y)$.

Since $\boldsymbol{\theta}^*$ is not known a priori, Soare et al. also proposed the $\mathcal{XY}$-Static allocation strategy for static arm allocation, where the arm selections $\mathbf{x}_t$ are fixed in advance and do not adapt to the rewards observed in previous rounds. This strategy pulls the arms to reduce uncertainty in all directions in $\mathcal{Y} = \{\boldsymbol{y} = \boldsymbol{x}_{a'} - \boldsymbol{x}_a : \forall a' \neq a\}$. The sample allocation is given by $\mathbf{x}_t = \arg\min_{\mathbf{x}_t} \max_{\boldsymbol{y}\in\mathcal{Y}} \|\boldsymbol{y}\|_{\boldsymbol{V}_{\mathbf{x}_t}^{-1}}$.

To improve over static design, Soare et al. (2014) proposed the $\mathcal{XY}$-Adaptive algorithm. This semi-adaptive method divides the rounds into phases, and uses static allocation within each phase. At the end of each phase, the algorithm eliminates uninformative directions. Importantly, this approach avoids the $\sqrt{d}$ term that appears in (3.18) for fully adaptive strategies.

To further improve performance, the Linear Gap-based Exploration (LinGapE) algorithm by Xu et al. (2018) employs a fully adaptive arm selection strategy. At each round, the learner finds the arm with the highest empirical mean i_t and the most ambiguous arm j_t, and then selects arm a_t that most reduces the confidence interval of the estimated gap between the two arms, $(\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})^\top \hat{\boldsymbol{\theta}}_t$. LinGapE determines the optimal allocation ratio $p_k^*(i_t, j_t)$ for each arm $k \in [K]$ as $t \to \infty$, and selects the arm whose sampling frequency is closest to this optimal ratio. A simpler alternative is the greedy rule, which selects the arm that most efficiently shrinks the confidence set along the critical direction. The two rules are given by,

$$a_{t+1}^{\text{Ratio}} = \arg\min_{a\in[K]: p_a^*(i_t,j_t)>0} \frac{T_a(t)}{p_a^*(i_t, j_t)}, \quad \text{and} \quad a_{t+1}^{\text{Greedy}} = \arg\max_{a\in[K]} \|\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t}\|_{(\boldsymbol{V}_t + \boldsymbol{x}_a\boldsymbol{x}_a^\top)^{-1}}, \tag{3.20}$$

where $T_a(t)$ denotes the number of times arm a has been pulled up to time t.

3.4.3.3 Fixed Confidence BAI for Unstructured Bandits

The unstructured case is the most extensively studied and forms the foundation for BAI in linear bandits. In particular, the lower bound in (3.19) is derived from the expected sample complexity established in (Garivier and Kaufmann, 2016). As this setting is not used in this thesis, we do not discuss it in detail. We refer the reader to the seminal works of Garivier and Kaufmann (2016) and Kaufmann et al. (2016). Extensions of this setting to distributed bandits have been proposed for both BAI and cumulative regret minimization (Li et al., 2023, Chen et al., 2023, Mitra et al., 2022, Réda et al., 2022).

3.4.4 Distributed BAI in Linear Bandits

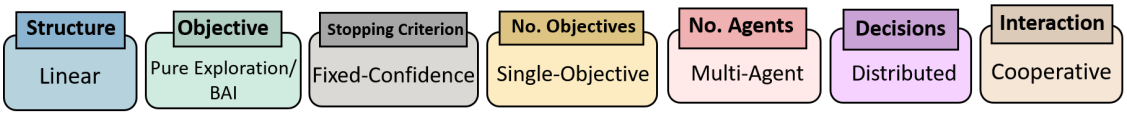


Figure 3.7: Distributed BAI in linear bandits in the fixed confidence setting.

The previous discussion focused on the single-agent setting, where all information available to an agent comes from the agent's own actions; this includes the arm selections and observed rewards. In contrast, the multi-agent setting enables collaboration among agents, allowing them to share information and accelerate the learning process by reducing the number of samples required per agent. However, communication between agents results in communication cost, and practical systems must balance between the overhead of frequent global updates and the inefficiency of excessive local updates.

Let M denote the number of agents. In the collaborative setting, each agent $m \in [M]$ observes its local rewards and actions, and also receives aggregated information from a central coordinator. Specifically, each agent maintains a local matrix update $\Delta \boldsymbol{A}_{t,m} = \sum_{s=t_B}^{t} \boldsymbol{x}_{a_{s,m}} \boldsymbol{x}_{a_{s,m}}^{\top}$ and vector update $\Delta \boldsymbol{b}_{t,m} = \sum_{s=t_B}^{t} \boldsymbol{x}_{a_{s,m}} r_{s,m}$ over the interval $[t_B, t]$, where t_B marks the beginning of the current global round. When a communication event is triggered, these updates are sent to the coordinator, which aggregates the data and returns the matrix $\boldsymbol{A}_{\text{coor},t} = \sum_{m=1}^{M} \Delta \boldsymbol{A}_{t,m}$ and vector $\boldsymbol{b}_{\text{coor},t} = \sum_{m=1}^{M} \Delta \boldsymbol{b}_{t,m}$. Each agent then uses these values to update its local confidence estimates in subsequent rounds.

In this thesis, we propose the Distributed LinGapE (DistLinGapE), a collaborative, adaptive, and synchronous algorithm for BAI in linear bandits in the fixed confidence setting that extends the single-agent LinGapE. We derive an upper bound on the per-agent sample complexity and the communication cost, and we apply the algorithm to solve the service placement problem in MEC.

While most existing works on distributed bandits focus on unstructured settings (Li et al., 2023, Chen et al., 2023, Mitra et al., 2022, Réda et al., 2022), works on linear bandits focus on minimizing the cumulative regret in synchronous (Wang et al., 2020) and asynchronous (He et al., 2022) scenarios, or in adversarial environments (Mitra et al., 2022). In contrast, our work addresses the distributed BAI problem in linear bandits under a fixed-confidence setting and applies it to the service placement problem in MEC.

3.4.5 Distributed BAI in the Fixed-Confidence Setting for MEC

We use the BAI algorithm in linear bandits in the fixed-confidence setting to address the problem of identifying the optimal service to place (cache) at the network edge in MEC. In this context, each edge server represents an agent, and each candidate service represents and arm. The reward obtained from selecting a service is measured by the reduction in latency achieved by completing the service in the edge compared to the cloud. Alternative performance metrics include service delay, operational cost, cache hit rate, or other relevant QoS indicators.

Since the MEC servers have access to the same set of services (arms), they can collaborate to identify the best service to place locally. Coordination and information exchange are managed by a macro base station serving as a centralized controller which aggregates local information and shares updates. This collaborative approach accelerates the learning process by reducing the number of sampling rounds required per server. However, this gain in sample efficiency comes at the cost of additional communication. This problem and the solution are detailed in Chapter 4.

3.5 Multi-Objective MAB

Many real-world problems are multi-objective (MO) in nature involving multiple, often conflicting goals. These problems typically have a set of non-dominated or Pareto optimal solutions, where improving one objective necessarily worsens at least one other. For instance, in network resource allocation, maximizing total throughput and ensuring user fairness are conflicting objectives since maximizing throughput can prioritize certain users and reduce system fairness.

The MO bandit problem is built upon the general framework of multi-objective optimization (MOO) (Miettinen, 1999), which aims to solve:

$$\text{maximize} \{f_1(\mathbf{v}), f_2(\mathbf{v}), \ldots, f_D(\mathbf{v})\}, \tag{3.21}$$

$$\text{subject to} \quad \mathbf{v} \in \mathcal{V}, \tag{3.22}$$

where each $f_j : \mathbb{R}^d \to \mathbb{R}$ is an *objective function*, and $D \geq 2$ is the number of objectives, which can be incommensurable (of different units). The decision variable vector

$\mathbf{v} = (\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_d)$ lies in the feasible region $\mathcal{V} \subset \mathbb{R}^d$. The corresponding objective vector is $\boldsymbol{\mu} = (\mu^1, \mu^2, \ldots, \mu^D)$, where $\mu^j = f_j(\mathrm{v})$ for $j \in [D]$. While the single-objective case focuses on the decision space, MO analysis focuses on the objective space. Solution approaches to MO-MAB problems are also based on the MOO literature (Miettinen, 1999) and will be discussed in the following sections.

In the MO-MAB setting, when the agent selects an arm at time t, it observes a *reward vector* $\boldsymbol{r}_t$, where $\boldsymbol{r}_t = (r_t^1, \ldots, r_t^j, \ldots, r_t^D)$, here, r_t^j is the reward for the j-th objective, and D is the total number of objectives. The mean reward vector for arm k is denoted by $\boldsymbol{\mu}_k = (\mu_k^1, \ldots, \mu_k^D)$, representing the expected rewards for all objectives.

The MO-MAB problem typically has multiple optimal arms, and we denote the set of Pareto-optimal arms by $\mathcal{A}^*$. In MO-MAB, performance is evaluated not only by the standard regret, measuring the gap between the expected reward of the selected arm and that of the optimal ones, but also by the *unfairness regret*, which quantifies the imbalance in sampling among the Pareto-optimal arms. Let $\mathbb{E}[T^*(t)]$ denote the expected number of times each optimal arm should be played by time t, and let $T_i^*(t)$ be the actual number of times an optimal arm $i \in \mathcal{A}^*$ has been selected up to time t. The unfairness regret is defined as (Drugan and Nowe, 2013):

$$R_{\mathrm{F}}(t) = \frac{1}{|\mathcal{A}^*|} \sum_{i \in \mathcal{A}^*} \left(\mathbb{E}[T^*(t)] - T_i^*(t)\right)^2, \tag{3.23}$$

where lower values of $R_{\mathrm{F}}(t)$ indicate a more balanced selection across the Pareto-optimal arms, thereby reflecting a fairer exploration strategy.

3.5.1 MO-MAB Regret Minimization

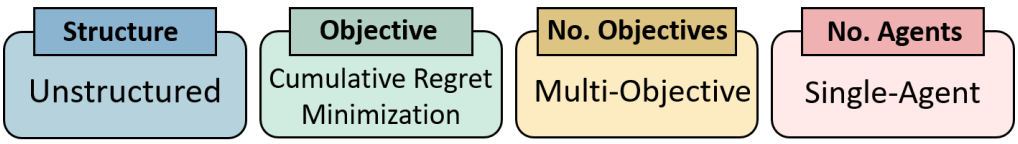


Figure 3.8: MO-MAB in the regret minimization setting.

Here we discuss two main approaches for solving the MO-MAB problem proposed by Drugan and Nowe (2013): The first relies on the partial ordering of the expected arm rewards, while the second method applies scalarization functions to transform reward vectors into scalars.

3.5.1.1 Partial Ordering

In this method, the set of optimal arms is found by comparing the mean rewards of the arms. For two arms with reward vectors $\boldsymbol{\mu}_a$ and $\boldsymbol{\mu}_b$, we have the following cases:

- $\boldsymbol{\mu}_a$ dominates $\boldsymbol{\mu}_b$, $\boldsymbol{\mu}_b \prec \boldsymbol{\mu}_a$, if and only if there is at least one dimension q for which $\mu_b^q < \mu_a^q$, and for all other dimensions z, $\mu_b^z \leq \mu_a^z$.
- $\boldsymbol{\mu}_a$ weakly dominates $\boldsymbol{\mu}_b$, $\boldsymbol{\mu}_b \preceq \boldsymbol{\mu}_a$, if and only if for all dimensions q, $\mu_b^q \leq \mu_a^q$.
- $\boldsymbol{\mu}_a$ is incomparable with $\boldsymbol{\mu}_b$, $\boldsymbol{\mu}_b \parallel \boldsymbol{\mu}_a$, if and only if there is at least one dimension q for which $\mu_b^q < \mu_a^q$ and there exists another dimension z for which $\mu_b^z > \mu_a^z$.
- $\boldsymbol{\mu}_a$ is non-dominated by $\boldsymbol{\mu}_b$, $\boldsymbol{\mu}_b \not\succ \boldsymbol{\mu}_a$, if and only if there is at least one dimension q for which $\mu_b^q < \mu_a^q$.

Fig. 3.9 illustrates these relationships using a bi-objective maximization example with eight arms. The non-dominated arms (marked with ×) are pairwise incomparable.

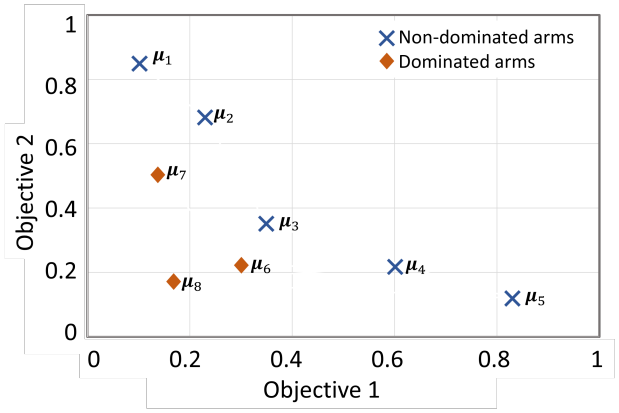


Figure 3.9: Partial ordering in a bi-objective problem.

Let $\mathcal{O}^*$ denote the set of Pareto optimal expected rewards, defined as the set of expected reward vectors that are non-dominated by any other vectors. The Pareto optimal set of arms, $\mathcal{A}^*$, is the set of arms whose expected reward vectors belong to $\mathcal{O}^*$.

The Pareto UCB1 algorithm (Drugan and Nowe, 2013) integrates the partial ordering of arms into the UCB algorithm. At each round t, the learner finds the UCB index for all arms as follows

$$\text{UCB}_k(t) = \hat{\boldsymbol{\mu}}_{t,k} + \sqrt{\frac{2\ln \sqrt[4]{D|\mathcal{A}^*|}}{T_k(t)}}, \tag{3.24}$$

where $\hat{\boldsymbol{\mu}}_{t,k}$ is the empirical mean reward vector for arm k, with $\hat{\mu}_{t,k} := \frac{1}{t}\sum_{s=1}^{t} r_{s,k}$, and $T_k(t)$ is the number of times arm k has been selected up to time t. Since the number of optimal arms $|\mathcal{A}^*|$ is unknown, it is replaced with $|K|$. The set of Pareto optimal arms at time t is computed based on the UCB indices. Then, an arm is sampled uniformly at random from this set and its reward is used to update the empirical mean. The algorithm

outputs the estimated Pareto optimal set $\mathcal{O}^*$.

The regret is defined as the distance between the expected reward vector of a suboptimal arm and the Pareto front. It is the smallest value $\epsilon_k > 0$, that when added to all objectives of a suboptimal arm k, the resulting (virtual) vector becomes incomparable to the optimal rewards in $\mathcal{O}^*$.

3.5.1.2 Scalarization Methods

Scalarization transforms reward vectors into scalars, allowing single-objective bandit algorithms to be applied in multi-objective problems. A widely used approach is *linear scalarization*, where the scalarized reward is defined as a weighted sum of the expected rewards of the different objectives of an arm (Drugan and Nowe, 2013, Drugan and Nowé, 2014). Since the learner typically cannot prioritize the objectives in advance, a set $\mathcal{S}$ of weight vectors is considered, each representing a different trade-off. For each weight vector $s \in \mathcal{S}$, the corresponding scalarization function is given by $f_{(s)}(\boldsymbol{\mu}_k) = \sum_{j=1}^{D} \omega_{(s)}^j \mu_k^j$, where $\left\{\omega_{(s)}^j\right\}_{j=1}^{D}$ are predefined weights satisfying $\sum_{j=1}^{D} \omega_{(s)}^j = 1$. Using these scalarization functions, single-objective bandit algorithms can be applied separately to each scalarization. For example, Drugan and Nowe (2013) proposed the Scalarized UCB1 algorithm that selects a scalarization function uniformly at random from $\mathcal{S}$ at the beginning of each round t, and then applies the classical UCB algorithm using the selected scalarization. The regret for the scalarization function s and arm a is $\max_{k\in[K]} f_{(s)}(\boldsymbol{\mu}_k) - f_{(s)}(\boldsymbol{\mu}_a)$.

However, linear scalarization cannot always find all optimal arms, especially in non-convex regions of the Pareto front. Therefore, alternative scalarization functions, such as the L_p-norm can be used as they can be better at finding all optimal arms (Drugan and Nowé, 2014).

3.5.2 MO-BAI in the Fixed Budget Setting

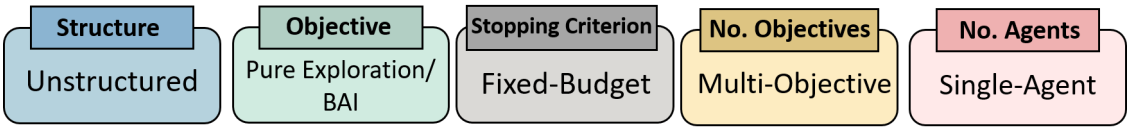


Figure 3.10: MO-MAB for BAI in the fixed budget setting.

Using scalarization methods, MO-BAI strategies can be developed from single-objective bandits. The scalarized successive rejects algorithm (Sabzehali et al., 2021) extends the successive rejects algorithm presented in Section 3.4.1.1. The main modification is that the total budget is divided across the number of scalarization functions. If the total budget

of the algorithm is n, then the budget allocated to each scalarization function is $n_S = \lfloor n/S \rfloor$, where S is the number of scalarization functions.

Additionally, the elimination process is performed for each scalarization function individually. That is, for each scalarization function, the algorithm starts with all arms as potentially optimal and performs the standard elimination process until a single arm remains. This arm is then added to the set of optimal arms, provided it has not already been included since an arm may be identified as optimal by multiple scalarization functions.

In Chapter 6, we extend the General Sequential Elimination algorithm to a cost-aware multi-objective setting. For each scalarization function and for each round t, both the reward vector and the cost of the selected arm are observed, and the optimal arm is defined as the one that maximizes the ratio of expected reward to expected cost. We derive the probability of arm misidentification under this framework.

3.5.3 MO-MAB for UAV Deplyment

In Chapter 6, we model the problem of deploying UAVs acting as base stations and FL clients as a MO-MAB problem. The UAVs adjust their coverage areas by selecting beamwidths from a discrete set of values, where each joint beamwidth configuration across the UAVs corresponds to an arm. Each configuration affects two values: (1) the number of covered sensors which we aim to maximize, and (2) the local processing delay which we aim to minimize. However, as processing delay increases with the number of covered sensors and collected data, the two objectives are conflicting. In addition, each configuration has an associated energy cost. The objective is to find the configuration that maximizes the ratio of the expected reward to the expected energy cost. To this end, we extend the General Sequential Elimination algorithm to the multi-objective case and consider the energy cost.

3.6 Mean Field Bandit Games

The previous sections focused on stochastic bandits where rewards are sampled from unknown but stationary distributions. This assumption does not hold in multi-agent settings, where due to agent interactions, the reward observed by an agent does not only depend on the chosen arm but also on the actions of other agents.

This section discusses multi-agent bandits, where multiple agents interact with the environment simultaneously and compete for shared resources. This setting is common in wireless communication systems and in MEC. For example, when multiple users access the same wireless channel or offload tasks to the same edge server, the resulting data rate or latency degrades as more agents choose the same option. This degradation in observed rewards due to increased competition is referred to as *negative externalities*.

We focus on scenarios with a very large number of agents, where computing game-

theoretic equilibria like the perfect Bayesian equilibrium becomes intractable. In such cases, mean field methods are used to approximate the overall dynamics of the agents. Under certain conditions, the distribution of agents across arms converges to a mean field steady state (MFSS), which appears stationary from the perspective of an individual agent. This enables modeling the system as a mean field MAB game, where each agent optimizes its actions assuming the population distribution across the arms remains nearly constant.

The mean field MAB game involves M agents, indexed by $m \in [M]$, each solving a MAB problem over discrete time steps $t = 1, 2, \ldots$, to select an arm k from a set of K arms with binary rewards (Gummadi et al., 2013). Each agent has

- **A type vector** that determines its rewards across arms. Each agent's type $\boldsymbol{\zeta}(m) \in \mathcal{Z} \subseteq [0,1]^K$ consist of agent-specific parameters that affect the reward obtained from each of the K arms. These type vectors are independently and identically distributed across agents, according to an unknown population measure W, where $W(\mathcal{Z})$ denotes the probability that $\boldsymbol{\zeta}(m) \in \mathcal{Z}$ (Gummadi et al., 2013). Both the agent's type and the population distribution W are unknown to the agent.

- **A state vector** determining the arm selections. The agent's state vector $\boldsymbol{\nu}_t(m) \in \mathbb{Z}_+^{2K}$ represents the cumulative number of wins and losses on each arm up to time t and is known to the agent. All agents follow a common policy $\pi : \mathbb{Z}_+^{2K} \to \mathcal{P}$, where $\mathcal{P} \subseteq [0,1]^K$ denotes a probability distribution over the arms, satisfying $\sum_{k=1}^{K} p_k = 1$.

Agents have geometric lifetimes. At each time step, an agent regenerates with probability $1-\beta$. Upon regeneration, its state $\boldsymbol{\nu}_t(m)$ is reset to zero and a new type $\boldsymbol{\zeta}(m)$ is drawn from W. Between regenerations, the agent retains its type.

The *population profile* at time t, denoted by $\boldsymbol{f}_t$, is a K-dimensional vector where $f_t(k)$ represents the number of agents playing arm $k \in [K]$. In the MFSS, the population profile converges to $\boldsymbol{f}$.

An agent pulling an arm k receives a reward of 1 with probability $Q(\zeta_{k,t}(m), f_t(k))$, where $\zeta_{k,t}(m)$ denotes the k-th component of the agent's type. The state vector is updated based on the observed reward. To ensure the existence of a MFSS, $Q(\zeta, \cdot)$ must be continuous in f. Rewards are conditionally independent across time steps, arms, and agents.

A UCB Policy Example : In non-regeneration rounds, an agent selects the arm maximizing the UCB index $\frac{w_k}{w_k+l_k} + \sqrt{\frac{\log(t)}{w_k+l_k}}$, where w_k and l_k are the number of wins and losses on arm k, respectively, since the last regeneration, and t is the time since regeneration. The selected arm is pulled, and a binary reward is observed with probability $Q(\zeta, f)$. The state is updated accordingly, and the process continues until the population profile reaches the MFSS.

3.6.1 Adjusting Mean Field Dynamics via Reward Scaling

The mean field MAB game naturally converges to a steady-state population profile $\boldsymbol{f}$, determined by the arm rewards and the interactions among agents. However, this equilibrium may not meet the practical system requirements, which often require a specific target profile $\boldsymbol{f}^*$ to ensure effective load balancing. To guide the system toward this target population profile, in Chapter 5, we propose a method that indirectly influences agents' decisions by scaling each arm's reward probability. These scaling factors adjust the incentive to select each arm, steering the collective behavior without requiring centralized coordination. This decentralized approach preserves the assumptions of the mean field model while allowing flexible control through local reward adjustments.

3.6.2 Application to Task Offloading

In Chapter 5, we consider a network with a very large number of users and a set of servers. Each user aims to offload its task to a server that can complete it within a specified expiry time. However, as more users access the same communication channel and server, the resulting delay increases. Since there is no central coordinator, users must make server selection decisions in a decentralized manner.

To address this challenge, we model the setting as a multi-agent bandit problem, where users are agents and servers represent arms. We solve it using a mean-field MAB game. Each user's type is determined by the quality of its communication channels to all servers, which directly impacts the offloading delay. The user's state vector tracks the number of successful and failed offloading attempts to different servers. We show that the mean-field MAB approach effectively solves the decentralized task offloading problem.

Additionally, we employ the proposed method for adjusting the population profile to balance the load across servers according to a predefined target distribution.

Part II

Scientific Contributions

Chapter 4

Service Placement in Small Cell Networks Using Distributed Best Arm Identification in Linear Bandits

Note: This manuscript was published as a journal paper in *IEEE Transactions on Mobile Computing* (Yahya et al., 2026). It is an extended version of a conference paper presented at EUSIPCO 2025 (Yahya et al., 2025), which is not included in this thesis.

Abstract As users in small cell networks increasingly rely on computation-intensive services, cloud-based access often results in high latency. Multi-access edge computing (MEC) mitigates this by bringing computational resources closer to end users, with small base stations (SBSs) serving as edge servers to enable low-latency service delivery. However, limited edge capacity makes it challenging to decide which services to deploy locally versus in the cloud, especially under unknown service demand and dynamic network conditions. To tackle this problem, we model service demand as a linear function of service attributes and formulate the service placement task as a linear bandit problem, where SBSs act as agents and services as arms. The goal is to identify the service that, when placed at the edge, offers the greatest reduction in total user delay compared to cloud deployment. We propose a distributed and adaptive multi-agent best-arm identification (BAI) algorithm under a fixed-confidence setting, where SBSs collaborate to accelerate learning. Simulations show that our algorithm identifies the optimal service with the desired confidence and achieves near-optimal speedup, as the number of learning rounds decreases proportionally with the number of SBSs. We also provide theoretical analysis of the algorithm's sample complexity and communication overhead.

4.1 Introduction

The rapid advancements in mobile communication technologies have profoundly transformed our society and reshaped sectors such as entertainment (Khan et al., 2022), health-

care (Liu et al., 2024), and transportation (Li et al., 2024). This transformation is accelerating with the widespread demand for 5G networks, which is expected to reach 6.3 billion users by 2030 (Ericsson, 2024), thereby increasing the need for innovative and scalable network infrastructures. One promising technology is mobile cloud computing, which allows mobile devices to offload computationally intensive tasks to a cloud equipped with large computation and storage capabilities, supporting complex applications. However, the network cloud is often placed too far from the end users, creating high network delays and increased traffic. To mitigate these issues, multi-access edge computing (MEC) has emerged as a promising paradigm that places computation and storage resources at the network edge, closer to end users (Wong et al., 2017). While MEC can significantly reduce latency and improve quality of service, it also introduces new challenges in resource allocation and service placement due to the limited edge resources, dynamic user behavior, and randomness of the environment.

The service placement problem addresses the challenge of determining which services to place at the resource-limited network edges. The services are typically hosted at the cloud, with only a subset placed at the edges, closer to the end users. When a user requests a service, the edge server responds if it hosts it. Otherwise, the task is offloaded to the cloud or other edge servers (Malazi et al., 2022, Ouyang et al., 2019). The benefits of effective service placement include reducing user-perceived delay, minimizing backhaul traffic, lowering service costs (Malazi et al., 2022), or meeting budget constraints (Chen and Xu, 2019). All these benefits depend largely on the demand (or popularity) of the services, which is not known a priori, making the problem of service placement quite challenging.

Some studies make the simplifying assumption that service demand, or its average value, is either known or can be directly estimated from recent observations (Chen et al., 2025, Farhadi et al., 2021, Yang et al., 2019a). However, this assumption does not accurately capture real-world service demand dynamics. To address this challenge, multi-armed bandit (MAB) approaches have been employed to learn service demand in an online manner (Chen and Xu, 2019, Chen et al., 2018, He et al., 2020). MAB is a sequential decision-making framework in which, in each round, the learner selects one of K arms (options) and receives a reward drawn from an unknown but fixed distribution. The most common MAB objective is to maximize cumulative reward or, equivalently, to minimize cumulative regret (Lattimore and Szepesvári, 2020). Another key objective is to identify the arm that maximizes the expected reward, a problem known as best arm identification (BAI), which falls under the pure exploration paradigm (Bubeck et al., 2009).

The contextual MAB (CMAB) framework has been applied to service placement as it leverages contextual information, such as service attributes or network conditions to enhance learning efficiency (Chen and Xu, 2019, Ouyang et al., 2019, Chen et al., 2018). Most existing works using standard MABs (Wang et al., 2023) or CMABs (Chen and Xu, 2019, Ouyang et al., 2019) in service placement focus on minimizing cumulative

regret, which requires balancing exploration (gathering information about new services) and exploitation (selecting the best-known service). While this approach is effective in dynamic settings where cost is measured by the obtained reward (Ouyang et al., 2019, Wang et al., 2023), it is less suitable for service placement problems, where the goal is to identify and deploy the best service for extended periods, often lasting hours (Chen and Xu, 2019). In such cases, the BAI framework is more appropriate, as it allows for an initial exploration phase to identify the best service, followed by long-term deployment.

In this chapter, we consider a small cell network consisting of a number of small base stations (SBSs) with computational resources, representing the edge servers in a MEC architecture. The network also includes a macro base station (MBS) that provides wide-area coverage for all users, forwards user service requests to the cloud, and facilitates information exchange among SBSs. We propose a distributed and adaptive multi-agent BAI algorithm in linear bandits that enables the SBSs to collaborate in identifying the best service placement with the desired confidence. Our approach reduces the number of learning rounds proportionally to the number of SBSs.

4.1.1 Chapter Contributions

In this chapter, we propose a distributed and adaptive multi-agent BAI algorithm in linear bandits. We then formulate the optimal service placement problem at the SBSs as a collaborative multi-agent BAI problem. Due to resource limitations at the SBSs, they can host only one service, while the remaining services are hosted in the cloud. Our objective is to identify the service that achieves the highest reduction in the total users' delay when placed at the SBSs instead of the cloud.

Since the delays and demand for different services is unknown, the SBSs collaborate to learn the delay improvement sequentially by leveraging contextual information that reflects the underlying structure of the bandit problem. Each SBS adapts its decisions according to past observations, enabling more informed actions in subsequent rounds. To accelerate this learning process, SBSs share data derived from their selected services and the corresponding observed rewards via the MBS. Because all SBSs access the same set of services and have the same bandit model, this collaboration significantly improves the overall learning efficiency. However, exchanging information across SBSs at every round can lead to excessive communication costs. To mitigate this, SBSs exchange information only when there is a significant change in their locally observed data, rather than in every round.

In summary, the contributions of this chapter are as follows:

- We propose a distributed and fully adaptive algorithm for multi-agent best arm identification in the linear bandit setting under a fixed-confidence framework.
- We formulate the service placement problem in small cell networks as a contextual linear bandit problem, aiming to identify the service that, when placed at the

SBSs, maximizes the reduction in the total user-perceived delay compared to cloud deployment.

- To the best of our knowledge, this is the first work to apply the BAI setting in linear bandits to a MEC problem, including service placement. We employ our proposed distributed BAI algorithm to determine the optimal service to deploy at the SBSs, leveraging collaboration among them to minimizes the number of learning rounds required by each SBS.
- We establish theoretical guarantees on the sample complexity and the number of communication rounds of the proposed algorithm.
- We analyze the robustness of the proposed algorithm to communication failures and show how it can be extended to heterogeneous agents.
- Numerical results demonstrate the effectiveness of our proposed algorithm in identifying the best arm (service) on both synthetic data and in the service placement problem. The algorithm identifies the best arm with the specified confidence while accelerating the learning process by nearly M-fold compared to independent learning by individual SBSs, where M denotes the number of SBSs (agents).

4.1.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 4.2 presents the related work. Section 4.3 introduces the system model for service placement in small-cell networks, while Section 4.4 formally defines the problem. Section 4.5 formulates the service placement problem as a distributed BAI problem and demonstrates how the proposed algorithm addresses it. In Section 4.6, we present our distributed BAI algorithm within the linear bandit framework. Section 4.7 shows how the proposed work can be extended or applied to heterogeneous settings and the limitations, while Section 4.8 analyzes the algorithm's robustness to communication failures. Section 4.9 provides numerical results, and Section 4.10 concludes the chapter and discusses future research directions.

4.2 Related Work

4.2.1 BAI in Linear Bandits

Linear bandits (LB) are a variant of MABs in which the learner observes a d-dimensional context vector for each selected arm and observes a reward that is a noisy linear function of the context (Abbasi-Yadkori et al., 2011). In this setting, the learner aims to estimate the unknown parameter vector that defines the linear relationship between the context and the expected rewards. The shared linear structure across arms enables the agent to learn

more efficiently compared to standard unstructured MABs, where the mean rewards of the arms are estimated independently. The standard stochastic MAB is a special case of LB when the set of available actions in each round is the standard orthonormal basis of $\mathbb{R}^d$.

In BAI, the learner sequentially samples arms with the objective of ultimately recommending a single best arm, that is, the one with the highest expected reward (Bubeck et al., 2009). BAI can be studied under two primary settings: fixed-budget and fixed-confidence. In the fixed-budget setting, the goal is to minimize the probability of misidentifying the best arm within a predefined number of rounds (Audibert and Bubeck, 2010). In the fixed-confidence setting, the objective is to identify the best arm with a specified level of confidence while minimizing the number of rounds required. In this setting, identification strategies are characterized by three fundamental components: the arm selection strategy, the stopping condition, and the arm recommendation rule (Kaufmann et al., 2016).

The strategies for BAI in LBs differ from those for conventional MABs because of the correlation between the arms. In LBs, pulling a suboptimal arm can still improve the estimation of the underlying linear model. The first algorithm for BAI in linear bandits appeared in Hoffman et al. (2014), but it is designed for the fixed budget case and does not take the complexity of the linear structure into account. Soare et al. (2014) were the first to address the BAI problem in LB under the fixed-confidence setting. They proposed the $\mathcal{XY}$-Static algorithm, which uses optimal experimental design for arm selection but suffers from high sample complexity due to its inability to adapt to the outcomes of the previous rounds. To enhance efficiency, they introduced the $\mathcal{XY}$-Adaptive strategy, which divides the total number of rounds into multiple phases. In each phase, a static allocation algorithm is used, but the strategy leverages the outcomes of previous phases to identify important directions and eliminate less promising ones. This approach can be considered semi-adaptive, as it does not adjust arm selection based on the results of each round. In contrast, Xu et al. (2018) introduced a fully adaptive algorithm, named Linear Gap-based Exploration (LinGapE), that dynamically adjusts the arm selection at each round based on all past observations, thus improving the sample complexity.

The standard MAB problem is extended to the distributed case where agents collaboratively maximize the cumulative regret (Wang et al., 2020, Shi and Shen, 2021, Chen et al., 2023, He et al., 2022) or identify the best arm for the pure exploration case (Mitra et al., 2021, Tao et al., 2019). The agents can access the same set of arms (Shi and Shen, 2021, Mitra et al., 2021, Huang et al., 2021a) or different subsets of arms (Chen et al., 2023), and in most cases they exchange information through a central coordinator. In addition to the objective of minimizing the cumulative regret or minimizing the sample complexity, distributed algorithms aim to minimize the communication cost whether it is measured in terms of the number of communication rounds (Mitra et al., 2021) or the amount of real numbers or bits transmitted (Wang et al., 2020, Amani et al., 2023, Huang et al., 2021a). Meeting both objectives is challenging because there is a tradeoff between

the number of local updates and the number of communication rounds. In elimination-based algorithms, for example, reducing local rounds can result in the elimination of the optimal arm, while reducing communication can increase the sample complexity.

In the literature, studies on distributed BAI in linear bandits for the fixed-confidence setting are still limited. In this chapter, we propose a distributed and adaptive multi-agent BAI algorithm in LB for the fixed confidence setting. The proposed algorithm is named Distributed Linear Gap-based Exploration (DistLinGapE) as it mainly extends the work in Xu et al. (2018) to the distributed setting. We also derive the sample complexity for the proposed algorithm and the upper bound on the number of communication rounds.

4.2.2 The Service Placement Problem

MEC problems are uncertain due to the randomness of the environment, user behavior, and the availability of network resources. Consequently, MAB-based strategies have been extensively applied in this domain (Chen and Xu, 2019, Ouyang et al., 2019, Chen et al., 2018, Yahya et al., 2024a, Zhang et al., 2017). In particular, the service placement problem in MEC is challenging due to the unknown service demand. To address this, several works have modeled it as a contextual bandit problem (Chen and Xu, 2019, Ouyang et al., 2019, Chen et al., 2018), with the modeling approach and solution strategy varying depending on the placement objectives and constraints. In Chen and Xu (2019), a service provider rents computational resources at edge servers to host application services, so it uses a CMAB approach to find the best placement that improves the users' QoS within the budget requirements. CMAB is also used in Chen et al. (2018) to find the optimal placement of a subset of services under a budget constraint first for non-overlapping SBSs, and then for overlapping coverage areas, adding difficulty to the problem as the service can be requested from multiple SBSs.

The work in Ouyang et al. (2019) considers the service placement problem from the users' perspective with the objective of jointly optimizing the user's perceived latency and service migration cost, weighted by user preferences. The problem is modeled as a CMAB problem where the context vector describes the users' state information such as location and service demand, and is solved using a Thompson-sampling-based algorithm. Contextual bandits are also used in content caching problems (Zhang et al., 2017); these problems differ from service placement, as they are concerned with accessing the cached data without processing.

Services deployed at the network edge typically remain in place for extended periods. Farhadi et al. (2021) propose a two-time-scale framework for service placement and request scheduling. In their approach, service placement decisions are made on a larger time scale, in terms of frames, while request scheduling is performed on a per-slot basis, depending on the real-time resource availability at each edge server. Their model assumes that the average demand for each service is known. This two-time-scale framework can

be particularly beneficial for energy-constrained devices, such as unmanned aerial vehicles (UAVs), where minimizing service caching overhead is critical (Zhou et al., 2022). These works highlight the importance of efficient long-term placement strategies and reinforce the relevance of our contribution, which uses BAI for long-term deployment while addressing collaboration and unknown service demand.

4.2.3 Algorithmic Choice for Service Placement

Service placement decisions are made on a slower timescale than request scheduling. Consequently, the objective is to identify, with high confidence, the service to be placed locally based on its long-term performance, which motivates a BAI formulation in linear bandits. Specifically, we adopt the LinGapE algorithm (Xu et al., 2018) as a fully adaptive arm selection approach and extend it to a collaborative multi-agent setting. Among existing BAI methods for linear bandits (Jedra and Proutiere, 2020, Jourdan and Degenne, 2022, Azizi et al., 2023), some provide strong theoretical guarantees and, in some cases, improved sample complexity. However, they often incur higher computational overhead or rely on assumptions that limit their applicability in distributed service placement systems. In contrast, LinGapE combines adaptivity with moderate computational complexity, making it well suited to our problem.

For example, the Lazy Track-and-Stop algorithm (Jedra and Proutiere, 2020) is asymptotically optimal, achieving the optimal sampling rate as the error probability $\delta \to 0$. However, for moderate values of δ, LinGapE can be more sample-efficient (Tirinzoni and Degenne, 2022). Moreover, Track-and-Stop requires solving a sample allocation optimization problem at each round, which results in significant computational overhead which motivated the introduction of a lazy variant. In our service placement problem, some level of error is acceptable and δ need not approach zero, especially when doing so incurs additional computational cost.

Other approaches focus on ε-best-arm identification, to identify any arm whose performance is within ε of the optimal one (Jourdan and Degenne, 2022). Although this relaxation can reduce sample complexity, extending ε-BAI to a distributed multi-agent setting is considerably more challenging as agents may observe different services as locally ε-optimal. This requires global consensus on arm elimination and coordinated sampling strategies across agents, thus increasing communication and problem complexity.

Meta-learning-based methods (Azizi et al., 2023) improve regret performance by transferring knowledge across multiple tasks under a fixed budget. This differs from our work, where agents collaboratively learn within a single bandit instance and jointly estimate the same underlying parameter vector over time.

Overall, LinGapE provides a balance between statistical efficiency, computational complexity, and scalability. Its extension to a distributed multi-agent framework enables cooperative service placement while maintaining a good tradeoff between learning speed

and communication cost, which is essential for practical deployments.

4.3 System Model

We first introduce the notation used throughout this work. **Notations:** We use boldface lowercase letters and boldface uppercase letters to represent vectors and matrices respectively. We use $[K]$ to represent a set of integers $\{1,\dots,K\}$. 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 term inside the brackets.

We consider a system comprising an MBS connected to the network cloud and M homogeneous SBSs. Each SBS serves as an edge server and provides communication coverage to users within its own non-overlapping small cell, resulting in M coverage areas of similar size. The cloud is equipped with large storage and computational resources, enabling it to host and process all user service requests forwarded by the MBS. The MBS provides wide-area coverage, collects user service requests, and relays them to the cloud. In contrast, the SBSs have significantly limited computational capabilities. They are connected to the MBS in a star-shaped network topology to enable information exchange. The system model is illustrated in Fig 4.1.

A service provider manages K distinct services that can be deployed either at the cloud or at the SBSs, depending on the SBSs' decisions. Each service $k \in [K]$ is characterized by a d-dimensional context vector $\boldsymbol{x}_k \in \mathbb{R}^d$. This vector can represent various factors such as the service type, computational and memory requirements, and geographic or time-based demand.

Here, our optimization problem depends on both the service demand and delay improvement of placing the service in the SBS instead of the cloud. Therefore, we construct the context vector as

$$\boldsymbol{x}_k = \tilde{G}_k \tilde{\boldsymbol{D}}_k, \tag{4.1}$$

where $\tilde{G}_k$ is the average delay improvement achieved when service k is placed at the SBS instead of the cloud for one request, and $\tilde{\boldsymbol{D}}_k \in \mathbb{R}^d$ is the long-term average service demand over d consecutive time periods. The duration of the time periods is selected based on the temporal dynamics of the network, where shorter periods allows the context to adapt to highly dynamic environments, while longer periods are more suitable for networks with stable or slowly varying demand. In our numerical results, $\tilde{\boldsymbol{D}}_k$ contains the users' average demand for service k over 8 time periods, representing a four-minute interval.

Due to limited edge resources, each SBS $m \in [M]$ can host only one service at any given time. If a user within the coverage area of SBS m requests a service that is unavailable at SBS m, it has to offload the request to the cloud via the MBS, incurring additional delay. The decision of which service to deploy at the SBSs depends on the objective of this placement. In this work, we aim to identify the service that reduces the total users'

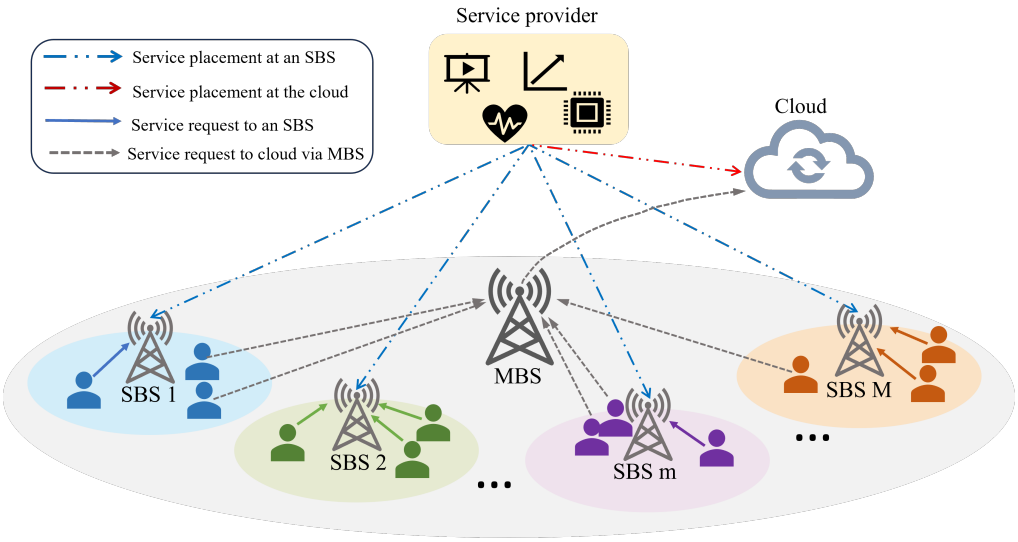


Figure 4.1: System model

delay the most when placed at the SBSs instead of the cloud. The key challenge arises from the uncertainty in delays and service demand, which are not known in advance.

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 delay improvement and effectively model the dependence of the service demand on the contextual information, we adopt the common approach of representing service demand as a noisy linear function of the context (Yang et al., 2018, Zhang et al., 2017). Mathematically, the demand for service $a_{t,m}$ at time t and SBS m is given by

$$p_{t,m} = \tilde{\boldsymbol{D}}_{a_{t,m}}^{\top} \boldsymbol{\omega}^* + \xi_{t,m}, \tag{4.2}$$

where $\boldsymbol{\omega}^* \in \mathbb{R}^d$ is an unknown parameter vector representing the underlying relationship between services' features and the user demands, and $\xi_{t,m}$ is the zero-mean R_ξ sub-Gaussian noise.

The users' delay depends on whether the requested service is placed at the SBS or the cloud, as detailed below.

4.3.1 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 (Chen et al., 2018). 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

$$\rho_{t,m} = W \log\left(1 + \frac{Ph_{t,m}}{I + \sigma_{\mathrm{N}}^2}\right), \tag{4.3}$$

where W is the uplink channel bandwidth, P is the users' 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. The round trip delay is $RTT_{t,m}$. Let s_k be the size of task k. Then, the communication delay of task k at SBS m is given by $d_{t,m}^{\mathrm{cm}}(k) = \frac{s_k}{\rho_{t,m}} + RTT_{t,m}$. The processing delay at SBS m is given by $d_{t,m}^{\mathrm{proc}}(k) = \frac{cs_k}{f_{t,k,m}}$, 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 and time t is:

$$d_{t,m}(k) = d_{t,m}^{\mathrm{cm}}(k) + d_{t,m}^{\mathrm{proc}}(k) = \frac{s_k}{\rho_{t,m}} + RTT_{t,m} + \frac{cs_k}{f_{t,k,m}}. \tag{4.4}$$

4.3.2 Service Delay at the Cloud

If the service requested by users in the service area of SBS m is not hosted at the SBS, the requests are offloaded to the cloud via the MBS, incurring transmission and processing delays. Let $\rho_{t,0}$ be the average data rate between the users and the MBS, where the index 0 is used for the cloud, indicating that the service request is relayed there. The wireless transmission delay is then given by $\frac{s_k}{\rho_{t,0}}$. Considering the delay caused by congestion and the routing policies in the backbone network between the MBS and the cloud server, the transmission delay over the backbone, with data rate ρ_t^{b}, is $\frac{s_k}{\rho_t^{\mathrm{b}}}$. Let $RTT_{t,0}$ denote the round trip delay. The overall communication delay is therefore $d_{t,0}^{\mathrm{cm}}(k) = \frac{s_k}{\rho_{t,0}} + \frac{s_k}{\rho_t^{\mathrm{b}}} + RTT_{t,0}$. The processing delay at the cloud is given by $d_{t,0}^{\mathrm{proc}}(k) = \frac{cs_k}{f_{t,k,0}}$, where $f_{t,k,0}$ is the CPU capacity allocated for service k at time t. The total service delay of task k at the cloud is thus given by

$$d_{t,0}(k) = d_{t,0}^{\mathrm{cm}}(k) + d_{t,0}^{\mathrm{proc}}(k). \tag{4.5}$$

4.3.3 Service Placement Utility

The utility of a service placement decision is defined as the improvement in the total user 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 average delay gain per request when service k is placed locally is modeled by the random variable $G_{t,m}(k)$, defined as $G_{t,m}(k) \coloneqq \mathbb{E}_u\left[d_{t,0}^u(k) - d_{t,m}^u(k)\right]$, where the expectation is taken over the users requesting service k.

If $p_{t,m}(a_{t,m})$ users request service $a_{t,m}$ at time t, the corresponding total delay improvement is $U_{t,m}(a_{t,m}) = G_{t,m}(a_{t,m})\, p_{t,m}(a_{t,m})$.

Both the average delay gain per request $G_{t,m}(k)$ and the service demand $p_{t,m}(k)$ are random. Since service placement decisions operate on a slow timescale, we approximate $G_{t,m}(k)$ by its long-term average

$$\tilde{G}_k \coloneqq \mathbb{E}[G_{t,m}(k)], \tag{4.6}$$

which is consistent with long-term service placement decisions and the use of the linear demand model. Substituting the demand model into the utility function, we obtain

$$U_{t,m}(a_{t,m}) \approx \tilde{G}_{a_{t,m}} \left(\tilde{\boldsymbol{D}}_{a_{t,m}}^{\top} \boldsymbol{\omega}^* + \xi_{t,m} \right) = \boldsymbol{x}_{a_{t,m}}^{\top} \boldsymbol{\omega}^* + \eta_{t,m} \tag{4.7}$$

where $\boldsymbol{x}_{a_{t,m}} \coloneqq \tilde{G}_{a_{t,m}} \tilde{\boldsymbol{D}}_{a_{t,m}}$ and $\eta_{t,m} \coloneqq \tilde{G}_{a_{t,m}} \xi_{t,m}$ is an independent, zero-mean, R-sub-Gaussian noise variable with $R = \max_{k \in [K]} |\tilde{G}_k| R_\xi$.

Therefore, the instantaneous delay improvement observed by SBS m can be expressed as a noisy linear reward

$$r_{t,m}(a_{t,m}) = \boldsymbol{x}_{a_{t,m}}^{\top} \boldsymbol{\theta}^* + \eta_{t,m}, \tag{4.8}$$

where $\boldsymbol{\theta}^*$ is an unknown parameter vector used to obtain the linear bandit formulation. Since the expected total delay improvement satisfies $\mathbb{E}[r_{t,m}(a_{t,m}) \mid \mathcal{F}_{t-1}, a_{t,m}] = \boldsymbol{x}_{a_{t,m}}^{\top} \boldsymbol{\theta}^*$, where $\mathcal{F}_{t-1}$ denotes the filtration up to time $t-1$, the same parameter vector governs the demand model and the reward model. Hence, for notational convenience, we write $\boldsymbol{\theta}^* = \boldsymbol{\omega}^*$.

4.4 Problem Formulation

Given the limited computation and storage resources at the SBSs, they aim to identify and deploy the service that yields the greatest reduction in the total users' delay compared to cloud deployment. Since the deployed service is typically used over an extended period, making an optimal selection with high confidence is crucial. Therefore, we formulate this problem as one where the SBS must identify the service that maximizes the expected utility with high confidence while minimizing the number of decision-making rounds. Let

$$a^* = \arg\max_{a \in [K]} \boldsymbol{x}_a^{\top} \boldsymbol{\theta}^* \tag{4.9}$$

be the optimal service that maximizes the expected utility. We aim to identify the estimated best service $\hat{a}_m^*$ such that

$$\mathbb{P}\left[(\boldsymbol{x}_{a^*} - \boldsymbol{x}_{\hat{a}_m^*})^{\top} \boldsymbol{\theta}^* > \varepsilon \right] \leq \delta \tag{4.10}$$

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., $\hat{a}^* = \hat{a}^*_m, \forall m \in [M]$. A key challenge in this problem arises from the uncertainty in service demand, which is unknown a priori.

4.5 Modeling Service Placement as a Distributed BAI Problem

In this section, we demonstrate that the service placement problem can be modeled as a distributed BAI problem within the linear bandit framework. We then solve it using the proposed DistLinGapE algorithm, presented in Section 4.6.

Each SBS $m \in [M]$ acts as an agent in the bandit problem, while each service $k \in [K]$ corresponds to an arm associated with a context vector $\boldsymbol{x}_k \in \mathbb{R}^d$ characterizing the service. Selecting an arm in the bandit framework corresponds to an SBS m hosting a service. Observing the reward at time t involves observing the total reduction in delay resulting from this service placement among all users in the coverage area of SBS m. The MBS acts as a central coordinator, facilitating information exchange among the SBSs.

The best service can be identified by iteratively learning the parameter vector $\boldsymbol{\theta}^*$ using the BAI algorithm. Since all SBSs share the same vector $\boldsymbol{\theta}^*$ and set of services, and can communicate through the MBS, collaboration between the SBSs speeds up the identification process compared to independent learning by individual SBSs.

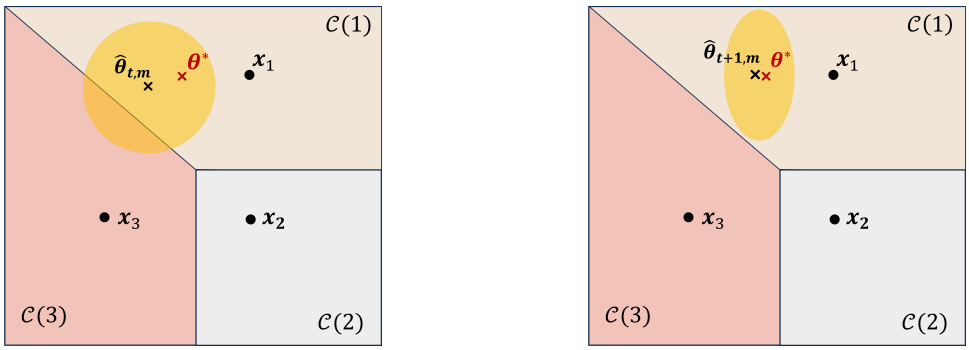
At a high level, the process of identifying the best service placement proceeds as follows. At time t, each SBS m hosts a service $a_{t,m}$ and observes the corresponding reward (utility). This reward is used to update the estimate of the unknown parameter vector, denoted by $\hat{\boldsymbol{\theta}}_{t,m}$, which guides the next service placement decision. To minimize the communication overhead, each SBS continues hosting services and updating its local information until a significant change is detected at one of the SBSs. When such a change occurs, a communication round is triggered, and the SBSs share their observations via the MBS. This process repeats until the best service is identified. Section 4.6 provides a detailed description of the proposed algorithm. To facilitate the explanation of the algorithm and the subsequent analysis, we will use the bandit terminology. The application to the small cell network follows simply by mapping each agent to an SBS and each arm to a service.

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

4.6 The DistLinGapE Algorithm

We consider a distributed LB problem with M agents, where $m \in [M]$, that can 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 $\boldsymbol{x}_k \in \mathbb{R}^d$, with $\|\boldsymbol{x}_k\| \leq L$ for some constant L. In round t, agent m pulls arm $a_{t,m} \in [K]$ and receives a reward that is a noisy linear function of the context vector as given by (4.8), where $\|\boldsymbol{\theta}^*\| \leq S$. The aim of our distributed algorithm is to enable the agents to collaboratively learn $\boldsymbol{\theta}^*$ to identify the best arm satisfying (4.10) by sharing observations over multiple communication rounds. Specifically, we aim to find the best arm with high confidence in the fewest number of rounds and a low communication overhead. The DistLinGapE algorithm is given in Algorithm 4.1.

Before discussing the algorithm, we give a high-level explanation of how the best arm is identified in linear bandits. Soare et al. (2014) show that for each arm, there exists a cone defining the set of $\boldsymbol{\theta}$ for which arm k is optimal, given by $\mathcal{C}(k) = \{\boldsymbol{\theta} \in \mathbb{R}^d, k \in \arg\max_{k \in [K]} \boldsymbol{x}_k^\top \boldsymbol{\theta}\}$. The optimal arm is identified when the confidence set of the estimate $\hat{\boldsymbol{\theta}}_{t,m}$ shrinks into one of the cones. The illustration in Fig. 4.2 gives an example with $K = 3$ arms and $d = 2$. The $\mathbb{R}^2$ Euclidean space is partitioned into three cones, and arm 1 is the optimal arm because $\boldsymbol{\theta}^* \in \mathcal{C}(1)$. In each round, the arm selection strategy aims to pull in the direction that speeds up the refinement of the confidence set (the yellow ellipsoid) to align with a cone. Fig. 4.2a shows the confidence set at time t. Since it overlaps with $\mathcal{C}(1)$ and $\mathcal{C}(3)$, arms 1 and 3 can be optimal. To shrink the confidence set into one cone only, the next arm pull must be in the direction $\boldsymbol{x}_3 - \boldsymbol{x}_2$. Fig. 4.2a shows the confidence set after the arm pull. Since the confidence set resides fully in cone 1, arm 1 is identified as the optimal.



(a) The confidence set at time t.

(b) The confidence set at $t+1$.

Figure 4.2: Illustration of BAI in LB.

4.6.1 Construction of Confidence Sets

In a single-agent linear bandit problem (Abbasi-Yadkori et al., 2011, Soare et al., 2014), the agent uses locally observed information to update its estimate of the parameter vector $\boldsymbol{\theta}^*$. Let λ denote the regularization parameter and $\boldsymbol{I}$ the identity matrix. The information includes the $d \times d$ design matrix $\boldsymbol{A}_t = \lambda \boldsymbol{I} + \sum_{s=1}^{t} \boldsymbol{x}_{a_s} \boldsymbol{x}_{a_s}^\top$, and the vector $\boldsymbol{b}_t = \sum_{s=1}^{t} \boldsymbol{x}_{a_s} r_s$. Using this, the agent computes the ℓ_2-regularized least squares estimate of $\boldsymbol{\theta}^*$, given by $\hat{\boldsymbol{\theta}}_t = \boldsymbol{A}_t^{-1} \boldsymbol{b}_t$.

However, in the collaborative algorithm, each agent m has access not only to its locally observed samples but also to the additional samples collected by the coordinator from all M agents. Let t_n denote the start time of the n-th communication round. The locally updated matrix at agent m at time t is $\Delta \boldsymbol{A}_{t,m} = \sum_{s=t_n}^{t} \boldsymbol{x}_{a_{s,m}} \boldsymbol{x}_{a_{s,m}}^\top$ and $\Delta \boldsymbol{b}_{t,m} = \sum_{s=t_n}^{t} \boldsymbol{x}_{a_{s,m}} r_{s,m}$. When an agent triggers a communication event, all agents share $\Delta \boldsymbol{A}_{t,m}$ and $\Delta \boldsymbol{b}_{t,m}$ with the coordinator, which aggregates them to compute

$$\boldsymbol{A}_{\text{coor},t} = \sum_{m=1}^{M} \Delta \boldsymbol{A}_{t,m} \text{ and } \boldsymbol{b}_{\text{coor},t} = \sum_{m=1}^{M} \Delta \boldsymbol{b}_{t,m}. \tag{4.11}$$

Both $\boldsymbol{A}_{\text{coor},t}$ and $\boldsymbol{b}_{\text{coor},t}$ are sent to the agents for use in the next communication round. Consequently, each agent uses the coordinator's aggregated data and its locally collected data to compute $\boldsymbol{A}_{t,m}$ and $\boldsymbol{b}_{t,m}$, defined in line 7 in Algorithm 4.1. Using this information, the ℓ_2-regularized estimate of the parameter vector is given as

$$\hat{\boldsymbol{\theta}}_{t,m} = \boldsymbol{A}_{t,m}^{-1} \boldsymbol{b}_{t,m}, \tag{4.12}$$

with the confidence bound given in Proposition 4.1 which is based on (Abbasi-Yadkori et al., 2011, Theorem 2).

Proposition 4.1 ***Confidence Ellipsoid*** *For linear bandits with conditionally R-sub-Gaussian noise for some $R \geq 0$, let $\lambda > 0$ and assume that $\|\boldsymbol{\theta}^*\|_2 \leq S$. Then, for any $\delta_m > 0$ the confidence set constructed from $\boldsymbol{A}_{t,m}$ for $t \in \{1, 2, \dots\}$ is*

$$\mathscr{C}_{t,m} = \left\{ \boldsymbol{\theta} \in \mathbb{R}^d : \|\hat{\boldsymbol{\theta}}_{t,m} - \boldsymbol{\theta}^*\|_{\boldsymbol{A}_{t,m}} \leq R \sqrt{2 \log \frac{\det(\boldsymbol{A}_{t,m})^{\frac{1}{2}}}{\lambda^{\frac{d}{2}} \delta_m}} + \lambda^{\frac{1}{2}} S \right\}. \tag{4.13}$$

Also, the following statement holds for $\boldsymbol{x} \in \mathbb{R}^d$

$$|\boldsymbol{x}^\top (\hat{\boldsymbol{\theta}}_{t,m} - \boldsymbol{\theta}^*)| \leq \|\boldsymbol{x}\|_{\boldsymbol{A}_{t,m}^{-1}} C_{t,m}, \tag{4.14}$$

where

$$C_{t,m} = R \sqrt{2 \log \frac{\det(\boldsymbol{A}_{t,m})^{\frac{1}{2}}}{\lambda^{\frac{d}{2}} \delta_m}} + \lambda^{\frac{1}{2}} S. \tag{4.15}$$

Lemma 4.1 *For any $\delta_m > 0$, $\boldsymbol{\theta}^*$ lies in the set $\mathscr{C}_{t,m}$ with probability $1 - M\delta_m$, for all $t \in \{1,2,\dots\}$ and all $m \in [M]$.*

***Proof* 4.1** *This follows from Proposition 4.1 and a union bound over all agents.*

From Lemma 4.1, we conclude that to ensure an output with an error of at most δ in (4.10) at each agent, we set $\delta = M\delta_m$.

4.6.2 The Arm Selection Strategy

The arm selection strategy at an agent aims to find the allocation $\mathrm{x}_t(m) = (\boldsymbol{x}_{a_{1,m}}, \dots, \boldsymbol{x}_{a_{t,m}})$ that identifies the best arm in the fewest number of rounds. To this end, at time t, each agent selects (without pulling) the arm with the maximum estimated reward, $i_{t,m}$, and the most ambiguous arm, $j_{t,m}$, given in lines 26 and 27 of Algorithm 4.1, respectively. Then, the agent pulls the arm that provides the largest information gain for estimating the gap of expected rewards $(\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}})^\top \boldsymbol{\theta}^*$. Geometrically, this shrinks the confidence set until it is contained within a single cone, thereby identifying the optimal arm (Xu et al., 2018).

In every round t of Algorithm 4.1, each agent m uses $\boldsymbol{A}_{t,m}$ and $\boldsymbol{b}_{t,m}$, given in line 7, to select the pulling direction specified by the SELECT-DIRECTION procedure. The estimated gap between arms $i_{t,m}$ and $j_{t,m}$ is

$$\hat{\Delta}_{t,m}(i,j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \hat{\boldsymbol{\theta}}_{t,m}, \tag{4.16}$$

and the confidence of the estimated gap is

$$\beta_{t,m}(i,j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{A}_{t,m}^{-1}} C_{t,m}, \tag{4.17}$$

where $C_{t,m}$ is given in (4.15).

In each time t, all agents check if the stopping condition is met, that is, if the upper confidence bound on the gap of the expected rewards falls below the target accuracy ε,

$$B_m(t) = \hat{\Delta}_{t,m}(j_{t,m}, i_{t,m}) + \beta_{t,m}(j_{t,m}, i_{t,m}) \leq \varepsilon. \tag{4.18}$$

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 and each agent selects arm $a_{t,m}$ that results in the highest decrease in $\|\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}}\|_{\boldsymbol{A}_{t,m}^{-1}}$ and consequently in the confidence bound $\beta_{t,m}(i_{t,m}, j_{t,m})$.

Following Soare et al. (2014), Xu et al. (2018), there are two approaches for the arm selection strategy. In the greedy approach, each agent pulls the arm that minimizes the confidence bound in the current round, given by

$$a_{t+1,m} = \underset{a \in [K]}{\arg\min} \|\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}}\|_{(\boldsymbol{A}_{t,m} + \boldsymbol{x}_a \boldsymbol{x}_a^\top)^{-1}}. \tag{4.19}$$

Although we do not analyze the theoretical guarantees of this approach, we empirically show that it performs well in the numerical results (Xu et al., 2018).

Alternatively, the second approach involves finding the optimal ratio for arm k appearing in the optimal allocation sequence $\mathrm{x}_t^*(m)$ as $t \to \infty$ (Xu et al., 2018). This ratio minimizes $\|\boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}}\|_{\boldsymbol{A}_{t,m}^{-1}}$. Let $w_k^*(i_{t,m}, j_{t,m})$ be the k-th element in the vector $\mathrm{w}^*(i_{t,m}, j_{t,m})$ defined as

$$\mathrm{w}^*(i_{t,m}, j_{t,m}) = \arg\min_{\mathrm{w}\in\mathbb{R}^d} |\mathrm{w}|, \tag{4.20}$$

$$\text{s.t.} \quad \boldsymbol{x}_{i_{t,m}} - \boldsymbol{x}_{j_{t,m}} = \sum_{k=1}^{K} w_k \boldsymbol{x}_k. \tag{4.21}$$

The optimal ratio for selecting arm k is given by:

$$p_k^*(i_{t,m}, j_{t,m}) = \frac{|w_k^*(i_{t,m}, j_{t,m})|}{\sum_{k=1}^{K} |w_k^*(i_{t,m}, j_{t,m})|}. \tag{4.22}$$

Let $T_{a,m}(t)$ denote the number of times arm a is selected up to time t. This includes the number of times arm a was selected by other agents in the past communication rounds and the number of times the arm is selected locally until time t. 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

$$a_{t+1,m} = \arg\min_{a\in[K]: p_a^*(i_{t,m}, j_{t,m})>0} \frac{T_{m,a}(t)}{p_a^*(i_{t,m}, j_{t,m})}. \tag{4.23}$$

Theorem 4.1 *The arm $\hat{a}_m^*$ returned by the DistLinGapE using the arm selection strategy in (4.19) or (4.23) satisfies (4.10).*

***Proof* 4.2** *The proof is presented in Appendix 4.B.*

4.6.3 DistLinGapE Sample Complexity

Using the arm selection strategy in (4.23), we derive an upper bound on the sample complexity of the proposed DistLinGapE algorithm and show that agent collaboration significantly reduces it.

The sample complexity is expressed in terms of the problem complexity, defined as follows (Xu et al., 2018)

$$H_\varepsilon := \sum_{k=1}^{K} \max_{i,j\in[K]} \frac{p_k^*(i,j)\alpha(i,j)}{\max\left(\varepsilon, \frac{\varepsilon+\Delta_i}{3}, \frac{\varepsilon+\Delta_j}{3}\right)^2}, \tag{4.24}$$

Algorithm 4.1 The DistLinGapE Algorithm

Input: ε, δ, S, R, ρ and λ
Output Arm $\hat{a}^*$ that satisfies stopping condition.
Initialize agents: $\forall m \in [M]$, $\boldsymbol{A}_{0,m} = \mathbf{0}$, $\boldsymbol{b}_{0,m} = \mathbf{0}$, $\Delta\boldsymbol{A}_{0,m} = \mathbf{0}$, $\Delta\boldsymbol{b}_{0,m} = \mathbf{0}$, $\Delta t_{0,m} = 0$. For strategy (4.23) initialize $\Delta T_{m,k} = 0, T_{m,k} = 0$ and $T_k = 0 \forall k \in [K]$
Initialize coordinator: Set $\boldsymbol{A}_{\text{coor},0} = \mathbf{0}$, $\boldsymbol{b}_{\text{coor},0} = \mathbf{0}$
for $t = 1, \ldots, \infty$ **do**
  for Agent $m = 1, \ldots, M$ **do**
    $\boldsymbol{A}_{t,m} = \lambda \boldsymbol{I} + \boldsymbol{A}_{\text{coor},t} + \Delta\boldsymbol{A}_{t,m}$, $\boldsymbol{b}_{t,m} = \boldsymbol{b}_{\text{coor},t} + \Delta\boldsymbol{b}_{t,m}$, and $\Delta t_{t,m} += 1$.
    Select pulling direction: $(i_{t,m}, j_{t,m}, B_m(t)) \leftarrow$ SELECT-DIRECTION$(\boldsymbol{A}_{t,m}, \boldsymbol{b}_{t,m})$
    if $B_m(t) \leq \varepsilon$ **then**
      $i_{t,m}$ is the best arm $\hat{a}^*$; Terminate algorithm
    end if
    Pull arm $a_{t,m}$ according to (4.19) or (4.23)
    Observe $r_{t,m}$
    Update $\Delta\boldsymbol{A}_{t,m} += \boldsymbol{x}_{a_{t,m}} \boldsymbol{x}_{a_{t,m}}^\top$,
    $\Delta\boldsymbol{b}_{t,m} += \boldsymbol{x}_{a_{t,m}} r_{t,m}$. For strategy (4.23) $\Delta T_{m,a_t}(t) += 1$ and $T_{m,a_t}(t) = T_{a_t}(t) + \Delta T_{m,a_t}(t)$
    // Communication condition
    if $\Delta t_{t,m} \log \frac{\det(\boldsymbol{A}_{t,m})}{\det(\lambda \boldsymbol{I} + \boldsymbol{A}_{\text{coor},t})} > D$ **then**
      Each agent $m \in [M]$ sends $\Delta\boldsymbol{A}_{t,m}, \Delta\boldsymbol{b}_{t,m}$, and $\Delta T_{m,k}(t), \forall k \in [K]$ (for strategy (4.23)) to coordinator.
      Agents reset $\Delta\boldsymbol{A}_{t,m} = 0, \Delta\boldsymbol{b}_{t,m} = 0$, $\Delta t_{t,m} = 0$, and $\Delta T_{m,k}(t) = 0$.
      Coordinator collects $\Delta\boldsymbol{A}_{t,m}$, $\Delta\boldsymbol{b}_{t,m}$ and $\Delta T_{m,k}(t) \forall k \in [K]$
      Coordinator computes $\boldsymbol{A}_{\text{coor},t} += \sum_{m=1}^{M} \Delta\boldsymbol{A}_{t,m}$,
      $\boldsymbol{b}_{\text{coor},t} += \sum_{m=1}^{M} \Delta\boldsymbol{b}_{t,m}$ and $T_k(t) = \sum_{m=1}^{M} \Delta T_{m,k}(t), \forall k \in [K]$
      Coordinator broadcasts $\boldsymbol{A}_{\text{coor},t}, \boldsymbol{b}_{\text{coor},t}$ and $T_k(t), \forall k \in [K]$ to all agents
    end if
  end for
end for

Procedure Select-Direction$(\boldsymbol{A}_{t,m}, \boldsymbol{b}_{t,m})$
  $\hat{\boldsymbol{\theta}}_{t,m} \leftarrow \boldsymbol{A}_{t,m}^{-1} \boldsymbol{b}_{t,m}$
  $i_{t,m} \leftarrow \arg\max_{i \in [K]} (\boldsymbol{x}_i^\top \hat{\boldsymbol{\theta}}_{t,m})$
  $j_{t,m} \leftarrow \arg\max_{j \in [K]} (\hat{\Delta}_{t,m}(j, i_{t,m}) + \beta_{t,m}(j, i_{t,m}))$
  $B_m(t) \leftarrow \max_{j \in [K]} (\hat{\Delta}_{t,m}(j, i_{t,m}) + \beta_{t,m}(j, i_{t,m}))$
  return $i_{t,m}, j_{t,m}, B_m(t)$
EndProcedure

where $\alpha(i,j) = |\mathrm{w}^*(i,j)|$ and $\Delta_i = (\boldsymbol{x}_{a^*} - \boldsymbol{x}_i)^\top \boldsymbol{\theta}^*$ for $a^* \neq i$. The value of H_ε captures

the order of the minimum number of samples required to distinguish between arms i and j with accuracy ε.

The improvement in sample complexity of the collaborative algorithm compared to the centralized (single-agent) algorithm is measured by the algorithm speedup. Let $T_{\mathcal{O}}$ be the per-agent sample complexity of the best centralized algorithm and $T_{\mathcal{A}}$ be the per-agent sample complexity of the proposed collaborative algorithm $\mathcal{A}$. The speedup is defined as (Tao et al., 2019)

$$S_{\mathcal{A}} = T_{\mathcal{O}}/T_{\mathcal{A}}. \tag{4.25}$$

The speedup can take any value between $1 \leq S_{\mathcal{A}} \leq M$. It is 1 if the agents learn in isolation, and M is the optimal speedup.

To understand the speedup $S_{\mathcal{A}}$, we note that in the multi-agent setting, the samples required to achieve an accuracy of ε in (4.24) are collected across M agents. Ideally, this would lead to a per-agent sample complexity that is $1/M$ of the single-agent case, i.e., $T_{\mathcal{A}} = T_{\mathcal{O}}/M$. However, due to the randomness in the observed rewards, which influences the triggering of communication events, the actual per-agent sample complexity can be higher, with $T_{\mathcal{A}} = T_{\mathcal{O}}/S_{\mathcal{A}}$.

Theorem 4.2 gives the upper bound on the per-agent sample complexity.

Theorem 4.2 *The DistLinGapE algorithm can identify the $(\varepsilon, M\delta_m)$-best arm using the arm selection strategy in (4.19) or (4.23) with probability $1 - M\delta_m$ and the following sample complexity per agent:*

- *For $\lambda \leq \frac{2R^2}{S^2} \log \frac{K^2}{\delta_m}$:*

$$\tau_m \leq \mu + 4\frac{H_\varepsilon}{M} R^2 \left(2\log\frac{K^2}{\delta_m} + d\log\left(1 + \frac{Y^2 L^2}{\lambda d}\right), \right) \tag{4.26}$$

  where $\mu = \frac{K}{M} + 1$, $Y = 2\sqrt{16 H_\varepsilon^2 R^4 d L^2/(M\lambda) + N^2}$, and $N = \frac{8 H_\varepsilon R^2}{M} \log \frac{K^2}{\delta_m}$.

- *For $\lambda > 4H_\varepsilon R^2 L^2$:*

$$\tau_m \leq 2\left(\frac{4H_\varepsilon R^2}{M} \log\frac{K^2}{\delta_m} + \frac{2H_\varepsilon \lambda S^2}{M} + \mu \right) \tag{4.27}$$

***Proof* 4.3** *The proof of Theorem 4.2 is given in Appendix 4.C.*

4.6.4 Communication Rounds

We measure the communication cost as the number of communication rounds in the learning process. To minimize this cost, an agent m initiates a communication round only when there is a significant change in the matrix $\boldsymbol{A}_{t,m}$. Since the volume of the confidence ellipsoid given by (4.13) depends on $\det(\boldsymbol{A}_{t,m})$, where $\det(\cdot)$ is the matrix determinant, this

change is evaluated by the ratio $\frac{\det(\boldsymbol{A}_{t,m})}{\det(\lambda \boldsymbol{I}+\boldsymbol{A}_{\text{coor},t})}$. Here, $\det(\lambda \boldsymbol{I}+\boldsymbol{A}_{\text{coor},t})$ reflects the volume at the beginning of the current communication round and $\det(\boldsymbol{A}_{t,m})$ is proportional to the volume of the confidence ellipsoid at round t (Wang et al., 2020).

Let τ be the total sample complexity of the DistLinGapE algorithm across all agents. The corresponding upper bound on the total number of communication rounds is given by Theorem 4.3.

Theorem 4.3 *The total communication cost of the DistLinGapE algorithm is upper bounded by* $O\left(M\sqrt{\frac{\tau d \log_2 \tau}{D}}\right)$.

Proof **4.4** *The proof of Theorem 4.3 is given in Appendix 4.D.*

4.6.5 Communication Threshold Selection

The communication threshold D in Algorithm 4.1 controls the tradeoff between communication overhead and convergence speed. Communication with the coordinator is triggered only when the condition in Line 15 of Algorithm 4.1 is satisfied.

Let T denote the (random) stopping time of DistLinGapE measured in global rounds, where a global round corresponds to an iteration at which communication with the coordinator may be triggered. Between two consecutive global rounds, each agent may perform multiple local updates and collect multiple samples. Let τ_m denote the total number of local samples collected by agent m up to stopping time T, and let $\tau = \sum_{m=1}^{M} \tau_m$ denote the total sampling complexity across all agents.

From Theorem 4.2, each agent has a deterministic upper bound $\tau_m \leq \bar{\tau}_m$, which implies $\tau \leq \sum_{m=1}^{M} \bar{\tau}_m$. Note that, in general, $\tau_m \geq T$, since agents may collect more than one local sample between communication events.

Since at most one synchronization event can be triggered per global round, and each synchronization consists of $2M$ transmissions, the total number of communications up to stopping time T satisfies

$$C(T) \leq 2MT \qquad \text{almost surely.} \tag{4.28}$$

We introduce a communication rate parameter $\rho \in (0,1]$ to control the communication overhead. The communication budget is defined as a fraction ρ of the number of global rounds and is given by

$$B_{\text{c}} = \lceil \rho\, T \rceil, \tag{4.29}$$

which corresponds, on average, to one synchronization every $1/\rho$ global rounds.

Using the communication bound in Theorem 4.3,

$$C(T) = O\left(M\sqrt{\frac{\tau d \log_2(\tau)}{D}}\right),$$

we choose the threshold D by inverting the bound so that the total number of communications satisfies $C(T) \leq B_{\mathrm{c}}$, which yields

$$D = \frac{M^2 \tau d \log_2(\tau)}{B_{\mathrm{c}}^2}. \tag{4.30}$$

With this choice of D, the total communication cost satisfies $C(T) = O(B_{\mathrm{c}})$ in the worst case.

4.6.6 Computational Complexity of DistLinGapE

The DistLinGapE runs in polynomial time, which makes it suitable for problems with a moderate number of arms and context dimension d. Consequently, it is well suited to our service placement problem with a moderate number of SBSs, services, and context dimensions.

The dominant computational cost of the algorithm comes from evaluating the confidence terms at the agents for arm selection, as discussed below.

Proposition 4.2 *Consider DistLinGapE with K arms and d-dimensional context vectors. By using the inverse matrix lemma to obtain the local inverse design matrix $\boldsymbol{A}_{t,m}^{-1}$, the computational complexity per round at each agent m is $O(Kd^2)$. Consequently, the total computational complexity is $O(M\tau_m Kd^2) + O\left(N_{comm}(Md^2 + MK)\right)$, where N_{comm} is the number of communication rounds.*

***Proof* 4.5** *At iteration t and agent m, DistLinGapE performs the following steps. First, it updates $\boldsymbol{A}_{t,m}^{-1}$ and $\hat{\boldsymbol{\theta}}_{t,m}$ using the inverse matrix lemma, which costs $O(d^2)$. Then, it identifies the current best arm which costs $O(Kd)$. Then, it identifies the most ambiguous arm by evaluating the confidence terms $\left\|\boldsymbol{x}_j - \boldsymbol{x}_{i_{t,m}}\right\|^2_{\boldsymbol{A}_{t,m}^{-1}}$, with a cost $O(d^2)$. Evaluating this quadratic form for all K candidate arms results in a total cost of $O(Kd^2)$, which dominates the other values. Hence, the computational complexity at each agent and each round is $O(Kd^2)$, and the cost over all iterations is $O(\tau_m Kd^2)$.*

In each global round, the coordinator aggregates the values of $\Delta\boldsymbol{A}_{t,m}$ at a cost of $O(Md^2)$, and $\Delta\boldsymbol{b}_{t,m}$ at a cost of $O(Md)$, and the $\Delta T_{m,k}, \forall k \in [K]$ at a cost $O(MK)$. So the aggregation cost over N_{comm} rounds is $O\left(N_{comm}(Md^2 + MK)\right)$.

4.7 Extensions to Heterogeneous Agents

The DistLinGapE algorithm proposed in Section 4.6 and the corresponding theoretical analysis consider a homogeneous multi-agent setting, where all agents have access to the same set of arms (services) and collaboratively identify a common best arm. This

model addresses an important cooperative service placement problem for SBSs operating in homogeneous environments, such as hospitals, industrial facilities, or campuses. It could be also used in more diverse environments with a subset of homogeneous SBSs, as collaboration among them accelerates learning. However, practical communication systems can be heterogeneous across agents, while sharing the same parameter vector $\boldsymbol{\theta}^*$.

Heterogeneity across SBSs can result from differences in user density or local demand patterns. This results in variations in context vectors, which affect the observed rewards of the arms. In this section, we discuss a heterogeneous case that can be addressed, to some extent, using the DistLinGapE algorithm. It considers agents with different sets of context vectors but a common reward ordering and share the same globally optimal arm. This setting models SBSs operating under different user densities but similar service preferences. Then, we discuss a possible extension for the DistLinGapE that considers agents with different contexts and potentially different reward orderings, where the objective is to identify the best arm locally at each agent, corresponding to SBSs with heterogeneous user preferences and densities.

4.7.1 Unequal User Density: Agents with a Common Arm Ordering

We consider the case in which SBSs may have different user densities but similar service preferences. Since the context vector used for service placement decision is constructed from long-term average service demand, the context vectors across SBSs differ only by a positive scaling factor. As a result, although the reward values vary across SBSs, the underlying arm ordering structure determined by the context vectors remains unchanged across agents.

In this setting, the DistLinGapE algorithm remains valid, as all agents have access to the same set of arms, corresponding to the same set of services, and the heterogeneity across SBSs is reflected only in the context vectors. To explain this, let the context vector of service k at SBS m be given by

$$\boldsymbol{x}_{k,m} = c_m \boldsymbol{x}_k, \quad c_m > 0,$$

where c_m reflects the user density at SBS m, and $\boldsymbol{x}_k$ represents the contextual information of service k. For any two services k and ℓ, the ordering of their expected rewards at SBS m is determined by

$$(\boldsymbol{x}_{k,m} - \boldsymbol{x}_{\ell,m})^\top \boldsymbol{\theta}^* \geq 0,$$

which is equivalent to

$$(\boldsymbol{x}_k - \boldsymbol{x}_\ell)^\top \boldsymbol{\theta}^* \geq 0.$$

As a result, the resulting partition of the parameter space into cones remains identical across all SBSs. Consequently, all M agents solve the same BAI problem in terms of space

partitioning and arm comparisons. The BAI process therefore follows Algorithm 4.1, with the scaling factors c_m affecting only the magnitude of the observed rewards and the rate at which confidence sets shrink, so the sample complexity and communication bounds derived for homogeneous contexts continue to hold, up to constant scaling factors.

4.7.2 Heterogeneous Demand: Local Best-Arm Identification

In a more general heterogeneous setting, SBSs may serve user populations with different service preferences, resulting in demand vectors that differ beyond simple scaling. Consequently, the reward ordering across services may differ across SBSs, even though rewards remain linear in the context with a shared parameter vector $\boldsymbol{\theta}^*$.

In this case, the context vectors associated with the arms (services) differ across agents and the separating hyperplanes defined by

$$(\boldsymbol{x}_{k,m} - \boldsymbol{x}_{\ell,m})^\top \boldsymbol{\theta} = 0$$

may no longer coincide across SBSs. This leads to different partitions of the parameter space into cones across agents and different local best arms. In this case, the learning objective shifts from identifying a single global best service to identifying the local best service at each SBS.

Despite this heterogeneity, collaboration in learning the shared $\boldsymbol{\theta}^*$ remains beneficial. By exchanging information, SBSs can improve the estimation accuracy of $\boldsymbol{\theta}^*$ and accelerate confidence shrinkage. However, the stopping and recommendation decisions must be defined locally at each SBS.

4.8 Robustness to Communication Failures

The proposed DistLinGapE algorithm assumes reliable communication; however, practical networks are prone to communication failures and delays. In these cases, the algorithm degrades gracefully. When updates between SBSs and the coordinator are delayed or unavailable, each SBS continues updating its local statistics $\boldsymbol{A}_{t,m}$ and $\boldsymbol{b}_{t,m}$ based on local observations. This reduction in information leads to slower convergence and increased sample complexity, while preserving the correctness of the stopping condition and the identification of the best arm. In the extreme case of full communication failure, the algorithm reduces to independent LinGapE learning at each SBS.

Lemma 4.2 *Consider the DistLinGapE algorithm under possible communication failures. For any SBS m, round t, and any pair of arms (i, j), the estimated gap under full communication and its confidence are given in (4.16) and (4.17), respectively. Let $\tilde{\beta}_{t,m}(i, j)$ be the confidence constructed using the potentially partial aggregated statistics*

available due to communication failures. Then, with probability at least $1-\delta$*, it holds that*

$$\tilde{\beta}_{t,m}(i,j) \;\geq\; \beta_{t,m}(i,j), \quad \forall t,\ \forall m \in [M],\ \forall i,j \in [K]. \tag{4.31}$$

Consequently, communication failures do not reduce the confidence bounds on the estimated gaps and therefore cannot cause early stopping or false identification of the best arm, they can only increase the stopping time.

***Proof* 4.6 (Proof sketch)** *At round* t*, let* $\boldsymbol{A}_{coor,t}$ *given in (4.11) be the aggregated updates at the coordinator under full communication. Under communication failures, the coordinator may only receive updates from a subset of SBSs, yielding*

$$\tilde{\boldsymbol{A}}_{coor,t} = \sum_{j \in \mathcal{S}_t} \Delta \boldsymbol{A}_{t,j}, \tag{4.32}$$

where $\mathcal{S}_t \subseteq \{1,\ldots,M\}$ *is the set of SBSs whose updates are successfully received by the coordinator at round* t*.*

Since each $\Delta \boldsymbol{A}_{t,j}$ *is positive semidefinite, then* $\tilde{\boldsymbol{A}}_{coor,t} \preceq \boldsymbol{A}_{coor,t}$*. The local design matrix constructed at SBS* m *under communication failures is* $\tilde{\boldsymbol{A}}_{t,m} = \lambda \boldsymbol{I} + \tilde{\boldsymbol{A}}_{coor,t} + \Delta \boldsymbol{A}_{t,m}$*. Therefore,* $\tilde{\boldsymbol{A}}_{t,m} \preceq \boldsymbol{A}_{t,m}$ *implies* $\tilde{\boldsymbol{A}}_{t,m}^{-1} \succeq \boldsymbol{A}_{t,m}^{-1}$*.*

For any pair of arms (i,j)*, it follows that* $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\tilde{\boldsymbol{A}}_{t,m}^{-1}} \geq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{A}_{t,m}^{-1}}$*, which implies*

$$\tilde{\beta}_{t,m}(i,j) \geq \beta_{t,m}(i,j). \tag{4.33}$$

Larger confidence bounds due to communication failures cannot satisfy the stopping condition early. Thus, communication failures may delay stopping but cannot lead to false identification of the best arm.

4.9 Numerical Results

In this section, we present and analyze the performance of the proposed algorithm, first using synthetic data and then through simulations of a small cell network. We compare the performance of the proposed DistLinGapE algorithm with the following baselines:

1. $\mathcal{XY}$**-Oracle Strategy** (Soare et al., 2014): This assumes that $\boldsymbol{\theta}^*$ is known to the learner and is used as a lower bound on the sample complexity.

2. $\mathcal{XY}$**-Adaptive Strategy** (Soare et al., 2014): It is a semi-adaptive algorithm that divides the total number of rounds into phases, and eliminates directions that do not contribute to reducing the uncertainty in estimating the parameter vector $\boldsymbol{\theta}^*$ at the end of each phase. Within each phase, a fixed allocation strategy is employed.

3. **LinGapE** (Xu et al., 2018): It is a fully adaptive approach that adjusts the arm selection based on the rewards observed in the previous round. This work is limited to the single-agent setting and serves as the main algorithm on which our work is based, extending it to the multi-agent case.

All results are averaged over 30 algorithm runs. The confidence δ_m is set to 0.05.

4.9.1 Simulation on Synthetic Data

Here we use the benchmark example from Soare et al. (2014), commonly used in BAI in LB. This example consists of $d+1$ arms, where d is the dimension of the context vector. 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.

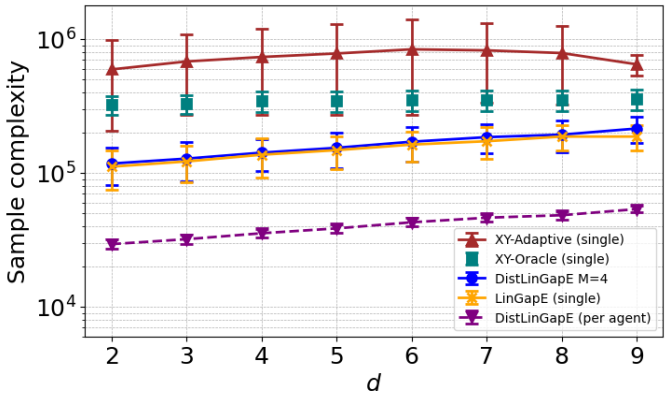


Figure 4.3: The sample complexity performance for different values of d.

Fig. 4.3 shows how the sample complexity changes with the change in d. The figure compares the proposed DistLinGapE algorithm with $M = 4$ agents with the single-agent (centralized) LinGapE, as well as the $\mathcal{XY}$-Oracle and the $\mathcal{XY}$-Adaptive algorithms. DistLinGapE, with $M = 4$ agents, achieves a total sample complexity close to that of LinGapE, effectively reaching the optimal speedup of M. On average, each agent contributes an equal number of arm pulls, resulting in a per-agent sample complexity of $1/M$ of the total sample count, as shown by the dashed line. In contrast, the $\mathcal{XY}$-Adaptive algorithm, being only semi-adaptive, does not use past observations at every time step. Consequently, it performs worse than fully adaptive algorithms, requiring a larger number of samples to achieve the same confidence level specified as $\delta = M\delta_m = 0.2$. Despite knowing $\boldsymbol{\theta}^*$, $\mathcal{XY}$-Oracle has higher sample complexity than DistLinGapE because DistLinGapE has tighter confidence bounds, which enable more accurate reward estimates (Xu et al., 2018).

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

Table 4.1 shows the number of samples per arm for $d = 5$. The DistLinGapE column represents the number of samples across all four agents. Arm 2 is pulled significantly more than any other arm, including the optimal arm 1. This occurs 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. Notably, arm 2 is nearly aligned with this direction, making it the most effective choice for reducing the uncertainty. The number of samples per agent in DistLinGapE is $1/M$ of the table value. In the service placement problem, this suggests that the server may repeatedly deploy a suboptimal service at the SBSs to confidently identify the best one.

4.9.2 Simulation on the Small Cell Network

We consider a network with a service provider with $K = 10$ services, which can be deployed either at the cloud or the SBSs. Once the optimal service deployment is identified, it remains fixed for an extended period compared to task scheduling. The context vector for each arm $\boldsymbol{x}_k$ is an 8-dimensional vector in this example, obtained from historical information about the the average delay improvement of a service and the long-term average demand over a four minute period. To model the demand vectors in our simulations, the base demand of each service follows a Zipf distribution and is perturbed by multiplicative log-normal noise to represent demand fluctuations across different time periods. Fig. 4.4 shows the normalized average service demand for the 10 services across 8 time periods.

To study the performance of the network under different numbers of collaborating SBSs, we consider a network consisting of six small cells, each with a radius of 200 m and an SBS located at its center. The cells are arranged in a circular layout such that their edges touch, forming an outer circle with a radius of 600 m. An MBS is positioned at the center of this outer circle. The value of M determines the number of SBSs collaborating in the learning process. Each small cell contains 75 users distributed uniformly at random within its coverage area. The main simulation parameters are summarized in Table 4.2, where parameters given as $[a,b]$ are sampled from the uniform distribution $\mathcal{U}[a,b]$.

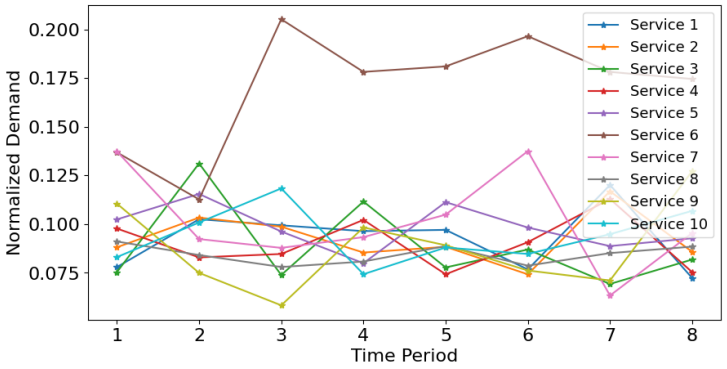


Figure 4.4: The normalized demand for the $K = 10$ services over 8 periods of time.

The uplink channel gain $h_{t,m}$ in (4.3) is defined as the average gain across users. In the simulation, the channel gain at time t for user u in SBS m is given by $h_{t,m,u} = \alpha^2_{t,m,u} \cdot g_m \cdot (l_{\text{ref}}/l_{t,m,u})^{\nu}$, where $\alpha^2_{t,m,u}$ represents the small-scale Rayleigh fading power gain, modeled as an exponential random variable with unit mean, and $g_m \cdot (l_{\text{ref}}/l_{t,m,u})^{\nu}$ models the path loss. Here, l_{ref} is the reference distance 1 m, $l_{t,m,u}$ is the distance between user u and SBS m, $g_m = -30$ dB is the path-loss coefficient, and $\nu = 2.5$ is the path-loss exponent. Similarly, the uplink channel between user u and the MBS is modeled as $\alpha^2_{t,0,u} \cdot g_0 \cdot (l_{\text{ref}}/l_{t,0,u})^{\nu}$, where $g_0 = -40$ dB, $\alpha^2_{t,0,u}$ models the small-scale Rayleigh fading power and follows an exponential distribution with mean 1, and $l_{t,0,u}$ is the distance between user u and the MBS.

Parameter	Description	Value
W	Channel bandwidth	10 MHz
P	Transmission power	10 dBm
σ^2_{N}	Noise power	-104 dBm
I	Interference power	-90 dBm
ρ^{b}	Backhaul data rate	$[1,4]$ GHz
$RRT_{t,m}$	Round trip time to SBS	$[2,7]$ ms
$RRT_{t,0}$	Round trip time to cloud	$[20,40]$ ms
s_k	task size	$[0.5,1]$ MB
$f_{t,k,0}$	Cloud CPU frequency	$[4.6,5.6]$ GHz
$f_{t,k,m}$	SBS CPU frequency	$[2.3,3.2]$ GHz
c	Processing cycles/bit	100

Table 4.2: Simulation parameters.

Fig. 4.5 shows the average delay reduction achieved by placing service k at the SBSs in a four SBS network, given that there are 75 users in each coverage area m. In this example, service 6 is the best. The DistLinGapE algorithm correctly identifies service 6

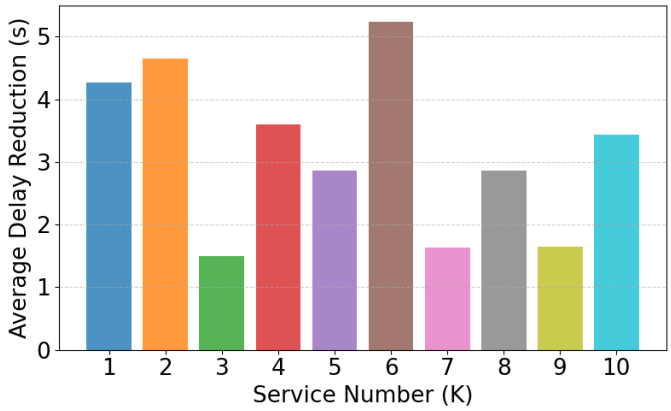


Figure 4.5: The average delay reduction for each service k for a network with $K = 10$ and $M = 4$.

as the best with zero error, despite the unknown service demand and the randomness of the environment.

Table 4.3 shows the effectiveness of collaborative learning in achieving near-optimal speedup. For each value M, we adjust the value of the communication threshold D as shown in the table to achieve a near-optimal speedup. The sample complexity of the centralized algorithm for a single SBS is $T_{\mathcal{O}} = 64631.15$ samples, which represents the minimum number of samples required to achieve the desired confidence level. If M SBSs learn independently, the total sample complexity would be $MT_{\mathcal{O}}$. Whereas the sample complexity of a centralized $\mathcal{XY}$-Oracle algorithm is 542901.40 samples. It is higher than that of the DistLinGapE, despite knowing $\boldsymbol{\theta}^*$, because the DistLinGapE has a tighter confidence bound, allowing for more accurate reward estimates.

When the SBSs collaborate, the DistLinGapE achieves a near-optimal speedup M. Table 4.3 shows the value of the communication threshold D and the number of communication rounds needed to achieve $\mathcal{S}_{\mathcal{A}}$ speedup in the simulation example with M collaborating agents.

M	1	2	4	6
D	∞	10	1	0.1
samples/agent	64631.12	30052.02	17051.46	10914.82
$S_{\mathcal{A}}$	-	1.86	3.79	5.92
Communication rounds	0	86.40	128.93	342.65

Table 4.3: Algorithm Speedup

Fig. 4.6 illustrates the impact of the communication threshold, D, on the number of

communication rounds performed by an agent and the total sample complexity required to achieve the desired confidence. The results correspond to a bandit instance with $K = 10$ services and $M = 4$ SBSs. Small values of D lead to excessive communication overhead, as agents exchange information too frequently, without improving the sample complexity. Conversely, selecting a large D increases the total sample complexity, as each agent performs a greater number of local updates before sharing information. This trade-off highlights the importance of carefully tuning D to balance communication efficiency and sample complexity.

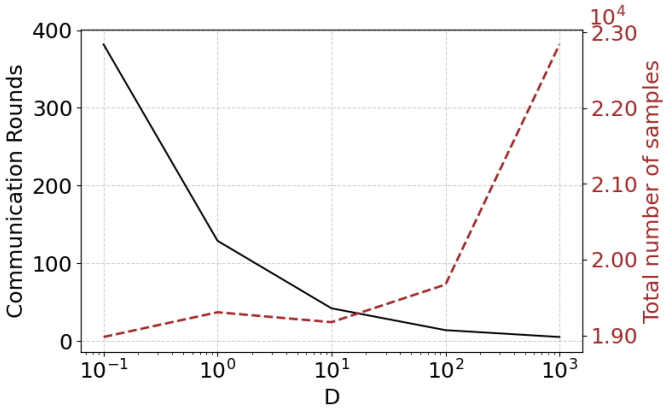


Figure 4.6: The communication threshold D vs. the number of communication rounds and the total sample complexity for a four-SBS network with $K = 10$ services.

Fig. 4.7 compares the sampling complexity and the number of communication rounds for a system with $M = 4$, $K \in \{10, 20, 30, 40, 50\}$, and 300 users per SBS. It shows that the sampling complexity is mainly affected by the gaps between the arms rather than by the number of arms itself. This explains why, in the figure, the sampling complexity at $K = 50$ is significantly lower than that at $K = 40$, for example. This behavior is consistent with the sampling complexity bound derived in Theorem 4.2, where the dependence on K appears only logarithmically, while the dominant factor is the set of arm gaps that determine the problem hardness H_ε. We note that, in this example, we increased the number of users per SBS from 75 to 300. With 75 users, the aggregated delay improvement over all users was too small, resulting in very slow confidence shrinkage and very long sampling times.

Fig. 4.8 and Fig. 4.9 show the learning behavior of DistLinGapE and centralized batch OFUL (Abbasi-Yadkori et al., 2011). For a fair comparison, since DistLinGapE samples M arms in each global round, a centralized M-batch OFUL is used to select the M arms with the highest indices. Both algorithms are run for 400 rounds with $D = 100$ for DistLinGapE. Fig. 4.8 shows that, under the same number of rounds and parallel samples, DistLinGapE achieves a higher cumulative delay reduction, indicating more effective sampling. Fig. 4.9 explains this behavior by presenting the average delay improvement per round. OFUL initially has a lower average delay improvement as it explores less certain services

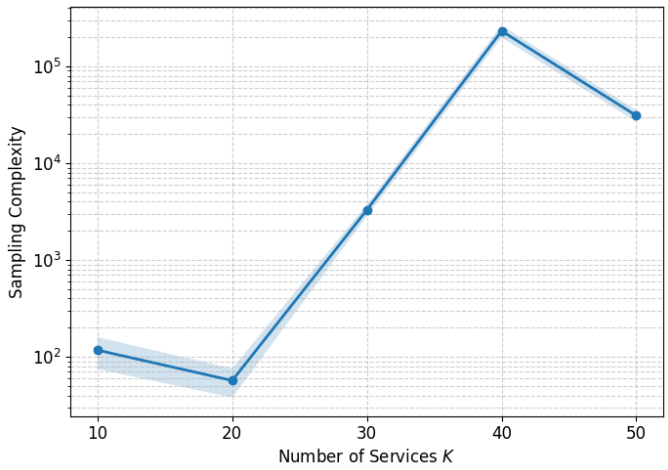


Figure 4.7: The sampling complexity and number of communication of the DistLinGapE for different number of services (arms).

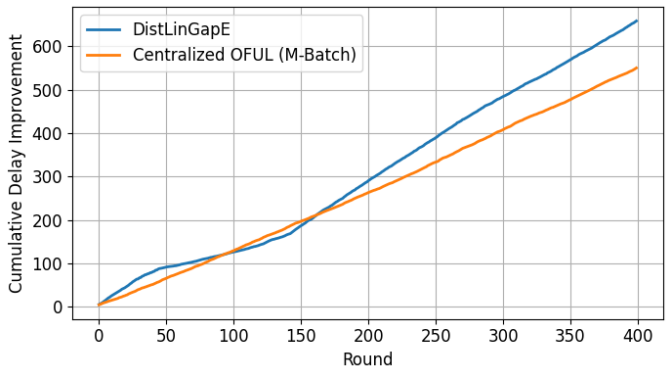


Figure 4.8: The cumulative delay improvement for the DistLinGapE with $M = 4$ SBSs and centralized OFUL with M-batch learning.

(least explored) based on its optimism sampling principle, whereas DistLinGapE concentrates sampling on services that are hardest to distinguish from the optimal one, enabling earlier sampling of delay efficient services.

4.10 Conclusion and Future Work

This work addresses the problem of service placement in small cell networks, considering the unknown service demand and the randomness of the environment. Given the limited resources at the SBSs, the goal was to identify the service that would result in the highest reduction in the total user delay when deployed at the SBSs instead of the cloud. To achieve this, we modeled the service demand as a linear function of service attributes

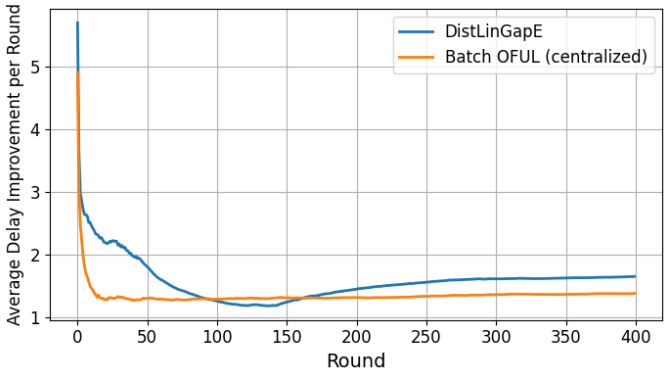


Figure 4.9: The average delay improvement per round for the DistLinGapE with $M = 4$ SBSs and centralized OFUL with M-batch learning.

and formulated the service placement problem as a distributed BAI problem. We then proposed a novel distributed multi-agent BAI algorithm for the fixed-confidence setting under the linear bandit framework. Our numerical analysis demonstrated the efficacy of the proposed approach in finding the optimal arm (service) and that collaboration between agents (SBSs) enables near-optimal speedup, reducing the sample complexity per agent by a factor of M compared to the case where the agents learn independently. Additionally, we derived upper bounds on both the sample complexity and the number of communication rounds required for the proposed algorithm. To the best of our knowledge, this work is the first to apply the BAI variant of MAB to the service placement problem. However, several directions remain for future research. One potential extension is to consider the top-m arms setting, where the algorithm identifies the best m arms instead of just one. This would allow service providers to deploy multiple services at SBSs, providing a more practical and flexible solution. Another promising direction is to extend this work to a heterogeneous settings with different contexts across agents. Furthermore, the overlap in coverage regions introduces coupling of requests, which must be addressed in future algorithms. Beyond algorithm design, theoretical guarantees, particularly upper bounds on sample complexity and communication rounds, need to be derived for these cases.

Appendix

4.A Auxiliary Lemmas

In this section, we present the lemmas necessary for proving the theorems stated in the chapter.

Lemma 4.3 *Let* $\Delta(i,j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{\theta}^*$ *be the gap of the expected rewards between arms*

i and j, and define a good event $\mathcal{E}_m$ for agent m as follows (Xu et al., 2018)

$$\mathcal{E}_m = \left\{\forall t > 0, \forall i, j \in [K], |\Delta(i,j) - \hat{\Delta}_{t,m}(i,j)| \leq \beta_{t,m}(i,j)\right\}. \tag{4.34}$$

The event $\mathcal{E}_m$ occurs with probability at least $1 - M\delta_m$.

***Proof* 4.7** *This lemma results from (4.14) in Proposition 4.1 followed by the union bound.*

The following lemma is based on (Xu et al., 2018, Lemma 5).

Lemma 4.4 *Under event $\mathcal{E}_m$ at agent m, $B_m(t)$ is bounded by*

$$B_m(t) \leq \min\left(0, -\max\left(\Delta_{i_{t,m}}, \Delta_{j_{t,m}}\right) + \beta_{t,m}(i_{t,m}, j_{t,m})\right) + \beta_{t,m}(i_{t,m}, j_{t,m}), \tag{4.35}$$

if either $i_{t,m}$ or $j_{t,m}$ is the best arm. Otherwise,

$$B_m(t) \leq \min\left(0, -\max\left(\Delta_{i_{t,m}}, \Delta_{j_{t,m}}\right) + 2\beta_{t,m}(i_{t,m}, j_{t,m})\right) + \beta_{t,m}(i_{t,m}, j_{t,m}). \tag{4.36}$$

Lemma 4.5 follows directly from (Xu et al., 2018, Lemma 2), adapted to the multi-agent setting by accounting for the number of times an arm is pulled, including the cumulative pulls by other agents in previous rounds as reported by the central coordinator.

Lemma 4.5 *Let $T_{m,i}(t)$ be the number of times arm i at agent m is sampled before round t, including the samples by other agents as reported by the central coordinator in the most recent communication round. Then, the norm $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{A}_{t,m}^{-1}}$ is bounded by*

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{A}_{t,m}^{-1}} \leq \sqrt{\frac{\alpha(i,j)}{T_{m,i,j}(t)}}, \tag{4.37}$$

where

$$T_{m,i,j}(t) = \min_{k \in [K]: p_k^*(i,j) > 0} T_{m,k}(t) / p_k^*(i,j). \tag{4.38}$$

Lemma 4.6 ***The Determinant-Trace Inequality (Abbasi-Yadkori et al., 2011, Lemma 10)*** *Suppose that $\boldsymbol{x}_{a_1}, \boldsymbol{x}_{a_2}, \ldots, \boldsymbol{x}_{a_t} \in \mathbb{R}^d$, for $1 \leq s \leq t$ and $\|\boldsymbol{x}_{a_s}\|_2 \leq L$. Let $\bar{\boldsymbol{A}}_t = \lambda \boldsymbol{I} + \sum_{s=1}^{t} \boldsymbol{x}_{a_s} \boldsymbol{x}_{a_s}^\top$ for some $\lambda > 0$. Then,*

$$\det(\bar{\boldsymbol{A}}_t) \leq (\lambda + tL^2/d)^d. \tag{4.39}$$

4.B Proof of Theorem 4.1

***Proof* 4.8** *Let arm $\hat{a}_m^*$ be the arm returned by agent m at the stopping time T. If this arm is worse than $\hat{a}^*$ by at least ε, $\Delta(a^*, \hat{a}_m^*) > \varepsilon$, then, by the definition of the stopping*

condition $B_m(T) \leq \varepsilon$ we get

$$\Delta(a^*, \hat{a}_m^*) > \varepsilon \geq B_m(T) \geq \hat{\Delta}_{T,m}(a^*, \hat{a}_m^*) + \beta_{T,m}(a^*, \hat{a}_m^*). \tag{4.40}$$

Using (4.34) and Lemma 4.3, the probability of returning a suboptimal arm at one of the agents is

$$\mathbb{P}[\Delta(a^*, \hat{a}_m^*) > \varepsilon] \leq \mathbb{P}[\bar{\mathcal{E}}_m] = 1 - \mathbb{P}[\mathcal{E}_m] \leq M\delta_m = \delta, \tag{4.41}$$

where $\bar{\mathcal{E}}_m$ is the complement of the event $\mathcal{E}_m$. This proves the condition in (4.10).

4.C Proof of Theorem 4.2

***Proof* 4.9** *Let agent m^* be the first agent to satisfy the stopping condition $B_{m^*}(t) \leq \varepsilon$. Also, let $\tilde{t}$ be the last round that arm k is pulled at agent m^*. Then, from Lemma 4.4, and because the stopping condition is not yet met, we have*

$$\begin{aligned} &\min\left(0, -\Delta_k + 2\beta_{\tilde{t}-1,m^*}(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*})\right) \\ &\quad + \beta_{\tilde{t}-1,m^*}(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*}) \geq B_m(\tilde{t}-1) \geq \varepsilon. \end{aligned} \tag{4.42}$$

In Lemma 4.4, $T_{m^*,i,j}(t)$ represents the total number of times arms i and j have been selected together up until time t. Due to collaboration, this value is not limited to the number of pulls at agent m^* but includes the pulls across all agents in past communication rounds.

By substituting the value of $\beta_{\tilde{t}-1,m^*} = \|\boldsymbol{x}_{i_{\tilde{t}-1,m^*}} - \boldsymbol{x}_{j_{\tilde{t}-1,m^*}}\|^{-1}_{\boldsymbol{A}_{\tilde{t}-1,m^*}} C_{\tilde{t}-1,m^*}$ in (4.42) and then using Lemma 4.5 we obtain

$$T_{m^*,i_{\tilde{t}-1},j_{\tilde{t}-1}} \leq \frac{\alpha(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*})}{\max\left(\varepsilon, \frac{\varepsilon+\Delta_{i_{\tilde{t}-1,m^*}}}{3}, \frac{\varepsilon+\Delta_{j_{\tilde{t}-1,m^*}}}{3}\right)^2} C^2_{\tilde{t}-1,m^*}, \tag{4.43}$$

where $C_{\tilde{t}-1,m^*}$ is defined in Proposition 4.1.

When arm k is pulled at $\tilde{t}$-th round at agent m^*,

$$T_{m^*,k}(\tilde{t}-1) = p^*_k(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*}) T_{m^*,i_{\tilde{t}-1},j_{\tilde{t}-1}}(\tilde{t}-1), \tag{4.44}$$

holds by definition. Therefore, the number of times arm k is pulled by the final round T,

$T_{m^*,k}(T)$, is bounded by

$$\begin{aligned}
&T_{m^*,k}(T) = T_{m^*,k}(\tilde{t}-1)+1 \\
&= p_k^*(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*}) T_{m^*,i_{\tilde{t}-1},j_{\tilde{t}-1}}(\tilde{t}-1)+1 \\
&\leq \max_{i,j\in[K]} p_k^*(i,j) T_{m^*,i_{\tilde{t}-1},j_{\tilde{t}-1,m^*}}(\tilde{t}-1)+1 \\
&\leq \frac{p_k^*(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*})\alpha(i_{\tilde{t}-1,m^*}, j_{\tilde{t}-1,m^*})}{\max\left(\varepsilon, \frac{\varepsilon+\Delta_{i_{\tilde{t}-1}}}{3}, \frac{\varepsilon+\Delta_{j_{\tilde{t}-1}}}{3}\right)^2} C^2_{\tilde{t}-1,m^*}+1 \\
&\leq \max_{i,j\in[K]} \frac{p_k^*(i,j)\alpha(i,j)}{\max\left(\varepsilon, \frac{\varepsilon+\Delta_i}{3}, \frac{\varepsilon+\Delta_j}{3}\right)^2} C^2_{T,m^*}+1. \qquad (4.45)
\end{aligned}$$

The total sample complexity across all agents is the sum of the number of samples across all arms, we have $\tau = \sum_{k=1}^K T_{m^*,k}(T)$, which yields

$$\tau \leq H_\varepsilon C^2_{T,m^*} + K, \qquad (4.46)$$

To find the per-agent sample complexity τ_m we note that $\tau_m = \lceil \tau/M \rceil$, so $\tau_m \leq \tau/M+1$. Using the upper bound in (4.46), we have

$$\tau_m \leq \frac{H_\varepsilon}{M} C^2_{T,m^*} + \mu, \qquad (4.47)$$

where $\mu = \frac{K}{M}+1$. Since the agents select arms sequentially, the sample complexity per agent is equal for all agents. Now, we find the upper bound on C^2_{T,m^*}.

Bounding the Confidence Term:

We extend the proof in Xu et al. (2018) to the multi-agent case. From Proposition 4.1 we have

$$C_{T,m^*} = R\sqrt{2\log\frac{K^2 \det(\boldsymbol{A}_{T,m^*})^{\frac{1}{2}} \det(\lambda \boldsymbol{I})^{-\frac{1}{2}}}{\delta_m}} + \lambda^{\frac{1}{2}} S. \qquad (4.48)$$

Using Lemma 4.6, and the fact that by the end of the algorithm at time T, the number of arm pulls is τ, we get

$$C_{T,m^*} \leq R\sqrt{2\log\frac{K^2}{\delta_m} + d\log\left(1+\frac{M\tau_m L^2}{\lambda d}\right)} + \lambda^{\frac{1}{2}} S, \qquad (4.49)$$

where we used the upper limit on the number of samples $\tau \leq M\tau_m$. To find an upper bound on the expression in (4.49), we consider two cases for λ:

Case 1: *For $\lambda \leq \frac{2R^2}{S^2}\log\frac{K^2}{\delta_m}$.*

In this case, (4.49) can be expressed as

$$C_{T,m^*} \leq 2R\sqrt{2\log\frac{K^2}{\delta_m} + d\log\left(1+\frac{M\tau_m L^2}{\lambda d}\right)}. \tag{4.50}$$

Substituting this value in (4.47),

$$\begin{aligned}\tau_m &\leq \frac{H_\varepsilon}{M}C_{T,m^*}^2 + \mu \\ &\leq 4\frac{H_\varepsilon}{M}R^2\left(2\log\frac{K^2}{\delta_m} + d\log\left(1+\frac{M\tau_m L^2}{\lambda d}\right)\right) + \mu. \end{aligned}\tag{4.51}$$

Let τ'_m be the parameter satisfying

$$\tau_m = 4\frac{H_\varepsilon}{M}R^2\left(2\log\frac{K^2}{\delta_m} + d\log\left(1+\frac{M\tau'_m L^2}{\lambda d}\right)\right) + \mu, \tag{4.52}$$

then, $\tau'_m \leq \tau_m$ holds.

Let $N = 8\frac{H_\varepsilon}{M}R^2\log\frac{K^2}{\delta_m} + \mu$, we have

$$\begin{aligned}\tau'_m \leq \tau_m &= 4\frac{H_\varepsilon}{M}R^2 d\log\left(1+\frac{M\tau'_m L^2}{\lambda d}\right) + N \\ &\leq 4\frac{H_\varepsilon}{M}R^2\sqrt{dL^2 M\tau'_m/\lambda} + N. \end{aligned}\tag{4.53}$$

By solving this inequality for τ'_m we obtain

$$\sqrt{\tau'_m} \leq 2\sqrt{16H_\varepsilon^2 R^4 dL^2/(M\lambda) + N^2}. \tag{4.54}$$

Let $Y = 2\sqrt{16H_\varepsilon^2 R^4 dL^2/(M\lambda) + N^2}$, then, using the upper bound of τ'_m in (4.52) yields

$$\tau_m \leq \mu + 4\frac{H_\varepsilon}{M}R^2\left(2\log\frac{K^2}{\delta_m} + d\log\left(1+\frac{Y^2 L^2}{\lambda d}\right)\right). \tag{4.55}$$

***Case 2**: For $\lambda > 4H_\varepsilon R^2 L^2$:*

Using the inequality $(a+b)^2 \leq 2(a^2+b^2)$, we can rewrite (4.49) as

$$C_{T,m^*} \leq 2\left(2R^2\log\frac{K^2}{\delta_m} + \frac{M\tau_m R^2 L^2}{\lambda} + \lambda S^2\right). \tag{4.56}$$

Substituting this value in (4.47) yields,

$$\begin{aligned}\tau_m &\leq \frac{H_\varepsilon}{M} C_{T,m^*} + K \\ &\leq 2\frac{H_\varepsilon}{M}\left(2R^2 \log\frac{K^2}{\delta_m} + \frac{M\tau_m R^2 L^2}{\lambda} + \lambda S^2\right) + \mu. \end{aligned} \tag{4.57}$$

Solving for τ_m, we get

$$\begin{aligned}\tau_m &\leq \left(1 - \frac{2H_\varepsilon R^2 L^2}{\lambda}\right)^{-1}\left(\frac{4H_\varepsilon R^2}{M}\log\frac{K^2}{\delta_m} + \frac{2H_\varepsilon \lambda S^2}{M} + \mu\right) \\ &\leq 2\left(\frac{4H_\varepsilon R^2}{M}\log\frac{K^2}{\delta_m} + \frac{2H_\varepsilon \lambda S^2}{M} + \mu\right). \end{aligned} \tag{4.58}$$

4.D Proof of Theorem 4.3

***Proof* 4.10** *In DistLinGapE, agents operate in parallel and are synchronized, meaning that they communicate with the central coordinator at the same time instants. We divide the running time into epochs, where each epoch ends with one communication round. Epoch p starts immediately after the p-th synchronization event and ends at the $(p+1)$ synchronization event. Let P denote the total number of epochs until stopping.*

Let T be the stopping time measured in global rounds. Let τ_m denote the total number of local samples collected by agent m up to stopping, and let $\tau := \sum_{m=1}^{M} \tau_m$ be the total sampling complexity across all agents. For each epoch p and agent m, let $\Delta\tau_{p,m}$ be the number of local samples collected by agent m during epoch p, so that

$$\sum_{p=0}^{P-1} \Delta\tau_{p,m} = \tau_m, \qquad \sum_{p=0}^{P-1}\sum_{m=1}^{M} \Delta\tau_{p,m} = \tau.$$

Let $\boldsymbol{A}_p$ denote the aggregated regularized information matrix at the start of epoch p, that is, $\boldsymbol{A}_p = \lambda \boldsymbol{I} + \sum_{\ell=0}^{p-1}\sum_{m=1}^{M} \Delta\boldsymbol{A}_{\ell,m}$, and let $\Delta\boldsymbol{A}_{p,m}$ be the local information matrix accumulated by agent m during epoch p. Hence, the aggregated matrix at the end of epoch p satisfies

$$\boldsymbol{A}_{p+1} = \lambda \boldsymbol{I} + \boldsymbol{A}_p + \sum_{m=1}^{M} \Delta\boldsymbol{A}_{p,m}.$$

Fix $\gamma > 0$ and define an epoch p to be long if $\max_{m\in[M]} \Delta\tau_{p,m} \geq \gamma$, and short otherwise. Since each long epoch satisfies $\max_m \Delta\tau_{p,m} \geq \gamma$, we have the number of long epochs N_{long} bounded by

$$N_{long} \leq \frac{1}{\gamma}\sum_{p=0}^{P-1} \max_m \Delta\tau_{p,m} \leq \frac{1}{\gamma}\sum_{p=0}^{P-1}\sum_{m=1}^{M} \Delta\tau_{p,m} = \frac{\tau}{\gamma}.$$

Now consider a short epoch p, for which $\Delta\tau_{p,m} < \gamma$ for all m. Communication at the end of epoch p is triggered by at least one agent; thus there exists an agent m_p such that

$$\Delta\tau_{p,m_p} \log\frac{\det\big(\boldsymbol{A}_p + \Delta\boldsymbol{A}_{p,m_p}\big)}{\det(\boldsymbol{A}_p)} > D. \tag{4.59}$$

Since $\Delta\tau_{p,m_p} < \gamma$, (4.59) implies

$$\log\frac{\det\big(\boldsymbol{A}_p + \Delta\boldsymbol{A}_{p,m_p}\big)}{\det(\boldsymbol{A}_p)} > \frac{D}{\Delta\tau_{p,m_p}} \geq \frac{D}{\gamma}. \tag{4.60}$$

Moreover, since $\sum_{m=1}^{M}\Delta\boldsymbol{A}_{p,m} \succeq \Delta\boldsymbol{A}_{p,m_p}$ and $\boldsymbol{A}_p \succ \boldsymbol{0}$, PSD monotonicity yields

$$\det(\boldsymbol{A}_{p+1}) = \det\Big(\boldsymbol{A}_p + \sum_{m=1}^{M}\Delta\boldsymbol{A}_{p,m}\Big) \geq \det\big(\boldsymbol{A}_p + \Delta\boldsymbol{A}_{p,m_p}\big),$$

and hence

$$\log\frac{\det(\boldsymbol{A}_{p+1})}{\det(\boldsymbol{A}_p)} \geq \log\frac{\det\big(\boldsymbol{A}_p + \Delta\boldsymbol{A}_{p,m_p}\big)}{\det(\boldsymbol{A}_p)} > \frac{D}{\gamma}. \tag{4.61}$$

Therefore, each short epoch contributes at least D/γ to the cumulative log-determinant growth, and number of short epochs N_{short}

$$N_{short}\cdot\frac{D}{\gamma} < \sum_{p=0}^{P-1}\log\frac{\det(\boldsymbol{A}_{p+1})}{\det(\boldsymbol{A}_p)} = \log\frac{\det(\boldsymbol{A}_P)}{\det(\boldsymbol{A}_0)}.$$

To bound the right hand side, note that $\boldsymbol{A}_P$ equals the aggregated information matrix at stopping and includes τ total rank-one updates across all agents. By the determinant–trace inequality in Lemma 4.6, we obtain

$$\det(\boldsymbol{A}_P) \leq \lambda^d\Big(1 + \frac{\tau}{\lambda d}\Big)^d,$$

and since $\boldsymbol{A}_0 = \lambda\boldsymbol{I}$,

$$\log\frac{\det(\boldsymbol{A}_P)}{\det(\boldsymbol{A}_0)} = \log\frac{\det(\boldsymbol{A}_P)}{\lambda^d} \leq d\log\Big(1 + \frac{\tau}{\lambda d}\Big) = O\big(d\log\tau\big),$$

where λ and d are fixed problem parameters. Hence,

$$N_{short} \leq \frac{\gamma}{D}\,d\log\Big(1 + \frac{\tau}{\lambda d}\Big).$$

We have $P = N_{\text{long}} + N_{\text{short}}$*, so*

$$P \leq \frac{\tau}{\gamma} + \frac{\gamma}{D}\, d \log\Big(1 + \frac{\tau}{\lambda d}\Big).$$

Minimizing the right hand side over $\gamma > 0$ *yields*

$$P = O\left(\sqrt{\frac{\tau\, d \log\big(1 + \frac{\tau}{\lambda d}\big)}{D}}\right) = O\left(\sqrt{\frac{\tau\, d \log \tau}{D}}\right). \tag{4.62}$$

Finally, each synchronization event consists of M SBSs to coordinator transmission and M coordinator to SBSs transmissions. Thus, the total communication cost is $C_{\text{tot}} = 2MP$*. Substituting (4.62) gives*

$$C_{\text{tot}} = O\left(M\sqrt{\frac{\tau\, d \log \tau}{D}}\right).$$

Chapter 5

Decentralized Task Offloading and Load-Balancing for Mobile Edge Computing in Dense Networks

Note: This work was published in *IEEE Communications Letters* (Yahya et al., 2024a).

Abstract We study the problem of decentralized task offloading and load-balancing in a dense network with numerous devices and a set of edge servers. Solving this problem optimally is complicated due to the unknown network information and random task sizes. The shared network resources also influence the users' decisions and resource distribution. Our solution combines the mean field multi-agent multi-armed bandit (MAB) game with a load-balancing technique that adjusts the servers' rewards to achieve a target population profile despite the distributed user decision-making. Numerical results demonstrate the efficacy of our approach and the convergence to the target load distribution.

5.1 Introduction

The stringent requirements of the emerging 5G/6G networks that necessitate low latency and high computational capabilities drove the evolution of mobile edge computing (MEC) technology. In MEC, computational and storage resources are strategically positioned near end users with limited resources in a communication network. Computation task offloading is an essential element of the MEC paradigm in which resource-limited devices offload their computation-intensive tasks for processing at edge servers with low computation and communication delays. Typically, MEC networks consist of a substantial number of users and servers, thereby raising the critical issue of effectively allocating communication and computation resources to optimize network performance. Another challenge is balancing the load among the network's edge servers to avoid prolonged service delays, which reduce the quality-of-service and decrease the system throughput (Ghomi et al., 2017). In some variations, computation costs or authorizations at differ-

ent servers may vary, imposing specific user loads on the servers. In centralized systems, a controller oversees incoming tasks and assigns them to servers, as exemplified by the Hadoop MapReduce software (Dsouza, 2015). Alternatively, in decentralized load-balancing, agents autonomously allocate their tasks to the servers (Liu et al., 2021). These algorithms can be either dynamic or static, depending on the current environment.

Optimal task offloading and load-balancing are further compounded in dynamic environments and in the absence of a centralized decision-maker. Therefore, solving the task offloading problem given precise network information, as in (Shi et al., 2023) among many others, is unrealistic in dynamic and uncertain environments. Consequently, some authors use online learning methods such as multi-armed bandits (MABs) to enable learning the network parameters and deciding about task offloading accordingly. References (Cheng and Maghsudi, 2023) and (Wang et al., 2022) apply multi-agent MABs for users with homogeneous and heterogeneous offloading requirements, where the reward considers the intrinsic task value and delay cost. Additionally, a budgeted MAB model can jointly minimize the delay and the energy cost (Ghoorchian and Maghsudi, 2021).

Compared to the state-of-the-art, our work offers several novelties. For example, (Cheng and Maghsudi, 2023, Wang et al., 2022, Ghoorchian and Maghsudi, 2021) assume equal resource availability for all devices and disregard potential competition over shared resources. Additionally, such approaches do not apply to dense networks due to excessive complexity. Furthermore, they can result in an undesired distribution of devices across the servers, thus degrading the network performance.

The mean field game (MFG) theory studies differential games with a large population of decision makers (Banez et al., 2021). Although its seminal model does not include uncertainty, MFG can be combined with learning to accommodate the lack of information. Reference (Gummadi et al., 2013) proposes an approach for analyzing an MAB game with numerous agents using a mean field framework. It shows that a non-stationary environment appears stationary by approximating the agents' interactions using their long-term average. This mean field MAB model was used in energy harvesting ultra-dense small cell networks to address the problem of decentralized user association based on their harvested energy amount and the population profile on the shared channels (Maghsudi and Hossain, 2017). In this chapter, we use the mean field MAB model in conjunction with load-balancing techniques for task allocation, mainly based on network delays.

We study dynamic decentralized task offloading and load-balancing in a dense network comprising several devices and a set of edge servers. The primary objective of each device is to receive the result of the task before an expiry time (Mancuso et al., 2023). This problem is complex as the delay is prone to high uncertainty due to the randomness in the communication channel, the task size, and the server's load. Additionally, since the devices share the wireless bandwidth and computation resources, the amount of resources allocated to each device depends on the actions of the others. To address this, we propose a novel approach: modeling the problem using multi-agent MABs, where each device is

an agent aiming to identify the best server. The reward obtained by an agent depends on the arm and the actions of other agents. Under such a scheme, it is unrealistic to analyze the performance of each agent individually, and so the agents' interactions are viewed as a dynamic game (Gummadi et al., 2013). To enhance this game, we propose an agent-based dynamic load-balancing algorithm that guides the agents to achieve a target distribution over the servers in the mean field steady state. This target distribution is determined centrally by the network manager based on the servers' heterogeneous costs or authorizations as the servers often belong to different operators. However, the task offloading decisions are decentralized (Mancuso et al., 2023) and thus scalable in dense networks thanks to less complex interactions between the devices and a lower cost of acquiring information than centralized methods.

5.2 System Model

We consider a dense network with n edge servers and m devices that offload heavy computational tasks of random sizes to the servers. The size of the task offloaded by device $k \in \{1,\dots,m\}$, denoted by s_k, follows a truncated normal distribution in the range $[s_a, s_b]$ (Chen et al., 2019). The devices compute tasks smaller than s_a locally, and s_b is the maximum task size that they can offload at once. Besides, to manage the dynamic nature of the problem, the devices use a time-slotted system to offload their computational tasks. At the beginning of a time slot t, each device k initiates the task offloading to an edge server. The channel between a device and server $i \in \{1,\dots,n\}$ is Rayleigh fading with bandwidth $B_{ki,t} = B/(mf_{i,t})$, where B is the bandwidth of the uplink channel, equal for all servers. $f_{i,t}$ is the fraction of devices that offload to server i at time t. The vector $\boldsymbol{f}_{i,t} = [f_{1,t},\dots,f_{n,t}]$ forms the population profile, which is the distribution of the devices on the servers. The size of the outcome of the task offloading process is proportional to the size of the offloaded task s_k, with proportionality constant ρ (Shi et al., 2023). The fading on the downlink and uplink channels is equal due to channel reciprocity, and the bandwidth of the downlink channel is constant. The edge servers have the same processing speed, each operating at a rate of F cycles/second. Each server can support running multiple tasks in parallel using virtualization technologies (Guo et al., 2024).

The delay experienced by a device k offloading to edge server i comprises three primary components: the uplink delay $d^{\mathrm{UL}}_{ki,t}$, the downlink delay $d^{\mathrm{DL}}_{ki,t}$, and the processing delay $d^{\mathrm{proc}}_{ki,t}$:

$$d_{ki,t} = d^{\mathrm{UL}}_{ki,t} + d^{\mathrm{DL}}_{ki,t} + d^{\mathrm{proc}}_{ki,t}. \tag{5.1}$$

1) Uplink Delay: This is the transmission delay of the offloaded task, and is inherently stochastic owing to the randomness in the communication channel and task size (Shi et al.,

2023):

$$d_{ki,t}^{\mathrm{UL}} = \frac{s_{k,t}}{B_{ki,t} r_{ki,t}} = \frac{s_{k,t}}{(B/mf_{i,t}) r_{ki,t}}, \tag{5.2}$$

where $r_{ki,t}$ is the spectral efficiency of the link between device k and edge server i, given by

$$r_{ki,t} = \log_2 \left(1 + \frac{p_0 |g_{ki,t}|^2}{I_{ki,t} + N_0} \right), \tag{5.3}$$

where g_{ki} is the parameter for the Rayleigh fading channel, implying that $h_{ki} = |g_{ki}|^2$ adheres to an exponential distribution. p_0 is the power transmitted by a given device, N_0 is the variance of the additive white Gaussian noise, and I_{ki} is the interference power following a uniform distribution (Xiao et al., 2020).[1]

2) Downlink Delay: It is the transmission time of the result of the offloaded task back to the user. Since the size of the result is very small compared to the size of the offloaded task, we assume that the results are transmitted over a fixed bandwidth B/ν. The delay is given by

$$d_{ki,t}^{\mathrm{DL}} = \frac{\rho s_{k,t}}{(B/\nu) r_{ki,t}}. \tag{5.4}$$

3) Processing Delay: Let $F_{ki,t} = F/(mf_{i,t})$ be the CPU cycles that edge server i allocates for the computational task offloaded by device k, and c be the number of processing cycles required for each bit of the offloaded task, then the task processing delay is (Shi et al., 2023)

$$d_{ki,t}^{\mathrm{proc}} = \frac{c s_{k,t}}{F_{ki,t}} = \frac{c s_{k,t}}{F/(mf_{i,t})}. \tag{5.5}$$

The task of device k is successfully offloaded to server i at round t if the total delay does not exceed $d_{\max}$,

$$d_{ki,t} = \frac{s_{k,t}}{B/(mf_{i,t}) r_{ki,t}} + \frac{\rho s_{k,t}}{\frac{B}{\nu} r_{ki,t}} + \frac{c s_{k,t}}{F/(mf_{i,t})} \leq d_{\max}. \tag{5.6}$$

5.3 Problem Formulation

The delays described in Section 5.2 are random due to the uncertainty in the offloaded task sizes and the channel fading. Additionally, the delays depend on the actions of the other devices due to the shared bandwidth and computation resources. The load distribution resulting from the decentralized nature of the problem might be undesirable. Therefore, this chapter aims at forcing a target device distribution on the servers. To formulate our

[1] An extension to mobile users and other fading types is straightforward if the signal-to-noise ratios are independent and identically distributed (i.i.d.) across devices.

problem, let $R_{ki,t}$ be the binary reward obtained by device k when offloading to server i,

$$R_{ki,t} = \begin{cases} 1 & d_{ki,t} \leq d_{\max} \\ 0 & d_{ki,t} > d_{\max} \end{cases}. \tag{5.7}$$

Also, denote the target population profile by $\boldsymbol{f}^*$. Each device k decides about offloading so as to maximize its cumulative reward over its lifetime T and achieve $\boldsymbol{f}^*$. Let A_t be the arm selected at round t, the optimization problem yields

$$\underset{A_t \in \{1,\ldots,n\}}{\text{maximize}} \quad \mathbb{E}\left[\sum_{t=1}^{T} R_{kA_t,t}\right], \tag{5.8}$$

$$\text{s.t.} \quad f_{i,t} = f^*_{i,t}, \quad \forall i \in \{1,\ldots,n\}. \tag{5.9}$$

5.4 The Mean Field Model

Before modelling the problem as a mean field game, we describe the mean field MAB game with binary rewards (Gummadi et al., 2013).

The game consists of *m agents*, each solving an MAB problem to select one of n available *arms* with unknown reward distributions. Upon selecting an arm, the agent receives a binary reward that depends on the number of other agents selecting the same arm. Additionally, each agent k has a *type* $\boldsymbol{\theta}_k \in [0,1]^n$ that is sampled from a probability distribution $W(\boldsymbol{\theta})$ during the generation of the agent. The ith element in $\boldsymbol{\theta}_k$ is a parameter affecting the agent's reward from arm i. The types are i.i.d. across the agents. Additionally, each agent has a known state $\boldsymbol{z}_{k,t} \in \mathbb{Z}_+^{2n}$ with $2n$ elements representing the agent's total number of wins and losses for each of the n arms over the agent's lifetime. The state in our task offloading problem is determined by the success in offloading a task from device k to server i in round t. If $d_{ki,t} \leq d_{\max}$ then the task is successfully offloaded (win) and element $(2i-1)$ of $\boldsymbol{z}_{k,t}$ is incremented by one. Otherwise, if $d_{ki,t} > d_{\max}$, the number of losses in element $2i$ is incremented by one.

The mean field model here assumes that all agents follow the same stationary *policy* to select an arm. Formally, let σ be the agent's policy, $\sigma : \mathbb{Z}_+^{2n} \to \{\boldsymbol{x} \in [0,1]^n : \sum_{i=1}^n x_i = 1\}$, where $\sigma(\boldsymbol{z}_{k,t}, i)$ denotes the probability that an agent chooses arm i when its current state is $\boldsymbol{z}_{k,t}$, $\sum_{i=1}^n \sigma(\boldsymbol{z}_{k,t}, i) = 1$. The policy used here is the upper confidence bound (UCB) (Gummadi et al., 2013).

To model the process of departure and arrival of agents, they are assumed to have a geometric lifetime, meaning that after each round they regenerate independently with probability $1-\beta$.

A key parameter in this model is the population profile, $\boldsymbol{f}_t = [f_{1,t}, f_{2,t}, \ldots, f_{n,t}]$, representing the fraction of agents playing each of the n arms at time t. The probability that

an agent k receives a unit reward when selecting an arm i, denoted by $Q(\theta_{ki}, f_i)$, is a function of the agent's type θ_{ki} and the fraction of agents selecting arm i, f_i. These rewards are independent across time, agents, and arms. Here, it is assumed that for a given θ_{ki}, $Q(\theta_{ki}, \cdot)$ is continuous in f_i.

Each time step t, every agent aims at solving its regret-minimization problem based on $\boldsymbol{z}_{k,t}$. Despite the interaction between agents, the system reaches the mean field steady state (MFSS) with stationary population profile $\boldsymbol{f}$ and state measure $\mu : \mathbb{Z}_+^{2n} \to \mathbb{R}^+$. The sufficient conditions for the existence of a unique MFSS are stated in (Gummadi et al., 2013, Theorem 1). In short:

Theorem 5.1 *Let L be the Lipschitz continuity constant. If the following conditions hold for all $a \in [0,1]$ and $x, x' \in [0,1]$:*

$$|Q(a,x) - Q(a,x^{'})| \leq L|x - x'|, \tag{5.10}$$

$$\beta(1+L) < 1, \tag{5.11}$$

where β is the continuation probability, then for any policy σ, there exists a unique fixed MFSS.

5.5 Task Offloading as A Mean Field MAB Problem

In this section, we model the distributed offloading problem as a mean field MAB game where the offloading devices are the agents, and the edge servers are the arms. We present our proposed load-balancing method in Section 5.6.

In this game, each agent k decides about offloading without any side information or inter-device communication. A round t corresponds to one of the following two cases:

Case 1: A regeneration round with probability $1-\beta$. The agent's state $\boldsymbol{z}_{k,t}$ is reset to zero and its type $\boldsymbol{\theta}_{k,t}$ is sampled from the distribution W. The ith element of the type is the normalized signal-to-interference-plus-noise-ratio (SINR):

$$\theta_{ki,t} = \frac{1}{\gamma_{\max}} \frac{p_0 h_{ki,t}}{I_{ki,t} + N_0}, \tag{5.12}$$

where $\gamma_{\max}$ is the maximum SINR to ensure that $\theta_{ki,t} \in [0,1]$.

Case 2: A continuation round with probability β. The agent's type does not change, and the agent runs the UCB policy that maps the current state vector to an action (arm selection). As a result, it observes a binary reward that depends on the arm's type and the fraction of users selecting arm i. To show this dependency, we rewrite the spectral efficiency in (5.3) as $r_{ki,t} = r(\theta_{ki,t}) = \log_2\left(1+\gamma_{\max}\theta_{ki,t}\right)$. The uplink rate of agent k to server i yields $r^{\mathrm{UL}}(\theta_{ki,t}, f_{i,t}) = \frac{B}{m f_{i,t}} r(\theta_{ki,t})$. The downlink data rate is given by $r^{\mathrm{DL}}(\theta_{ki}) = (B/\nu) r(\theta_{ki})$. The total delay at time t is calculated as follows, where we remove the

subscript t for simplicity:

$$d(\theta_{ki}, f_i) = \frac{cms_k f_i}{F} + \frac{s_k}{r^{\mathrm{UL}}(\theta_{ki}, f_i)} + \frac{\rho s_k}{r^{\mathrm{DL}}(\theta_{ki})}. \tag{5.13}$$

The binary reward for agent k for offloading a task to server i is 1 if $d(\theta_{ki}, f_i) \leq d_{\max}$ and zero otherwise. The cumulative density function (CDF) of the data size distribution determines the success probability $Q(\theta_{ki}, f_i)$. Given that the packet sizes follow a truncated normal distribution with density proportional to $\mathcal{N}(\mu_{\mathrm{s}}, \sigma_{\mathrm{s}})$ and $s_k \in [s_a, s_b], s_a{>}0$, the CDF yields

$$\begin{aligned} Q(\theta_{ki}, f_i) &= \mathbb{P}\left[d(\theta_{ki}, f_i) \leq d_{\max}\right] \\ &= \mathbb{P}\left(s_k \leq \frac{d_{\max} FBr(\theta_{ki})}{mf_i\left(cBr(\theta_{ki}) + F\right) + \rho \nu F}\right) \\ &= \frac{\Phi'\left(\frac{d_{\max} FBr(\theta_{ki})}{mf_i(cBr(\theta_{ki})+F)+\rho\nu F}\right) - \Phi'(s_a)}{\Phi'(s_b) - \Phi'(s_a)}, \end{aligned} \tag{5.14}$$

where $\Phi'(.)$ is the CDF of the standard normal distribution with its argument normalized by μ_s and σ_s, $\Phi'(x) = \frac{1}{2}\left(1{+}\mathrm{erf}\left(\frac{x-\mu_s}{\sqrt{2}\sigma_s}\right)\right)$.

Proposition 5.1 *The task allocation mean field MAB game has a unique MFSS.*

Proof. Theorem 5.1 states that a unique fixed MFSS exists if the conditions in (5.10) and (5.11) hold. For a type $\theta_{ki} \in [0,1]$ and fractions $f_i, f_i' \in [0,1]$ of agents, (5.10) is expressed as $|Q(\theta_{ki}, f_i) - Q(\theta_{ki}, f_i')| \leq L|f_i - f_i'|$.

As $Q(\theta_{ki}, f_i)$ is differentiable on [0,1], we prove that it is Lipschitz continuous by showing that $\left|\frac{\partial Q(\theta_{ki}, f_i)}{\partial f_i}\right| \leq L$ as follows:

$$\left|\frac{\partial Q(\theta_{ki}, f_i)}{\partial f_i}\right| = \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{s}} Z} \frac{md_{\max} FBr(\theta_{ki})\left(cBr(\theta_{ki}){+}F\right)}{\left(mf_i\left(cBr(\theta_{ki}){+}F\right){+}\rho\nu F\right)^2} e^{-\frac{1}{2\sigma_{\mathrm{s}}^2}\left(\frac{d_{\max} FBr(\theta_{ki})}{mf_i\left(cBr(\theta_{ki})+F\right)+\rho\nu F} - \mu_{\mathrm{s}}\right)^2}, \tag{5.15}$$

where $Z = \Phi'(s_b) - \Phi'(s_a)$. The exponential term is bounded, $\left|e^{-\frac{1}{2\sigma_{\mathrm{s}}^2}\left(\frac{d_{\max} FBr(\theta_{ki})}{mf_i\left(cBr(\theta_{ki})+F\right)+\rho\nu F} - \mu_{\mathrm{s}}\right)^2}\right| \leq$ 1, and the first term is maximized when $f_i = 0$. Therefore, the Lipschitz constant satisfies $L = \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{s}} Z} \frac{md_{\max} FBr(\theta_{ki})(cBr(\theta_{ki})+F)}{(\rho\nu F)^2}$. Now, by condition (5.11) the MFSS is unique when $\beta(1{+}L) < 1$. □

5.6 Load-Balancing

In this section, we develop a simple load-balancing algorithm that enables the network manager to change the agents' distribution over the servers in a decentralized manner.

In Section 5.5, we showed that the offloading problem can be interpreted as a stationary mean field MAB game and proved that a unique MFSS exists for this game. That means, if all parameters remain fixed, the population profile $\boldsymbol{f}_t$ converges to a fixed vector $\boldsymbol{f}$ after a sufficiently large number of rounds t. However, the profile $\boldsymbol{f}$ might be undesirable due to practical limitations, such as the servers' cost or accessibility. Thus, we now consider a general form of load-balancing, where we change the arms' rewards such that the resulting population profile of the game closely matches a desired profile $\boldsymbol{f}^*$. We do this by multiplying the reward probabilities $Q(\theta_{ki}, f_i)$ by arm-specific values $\alpha_i \in [0,1]$. This can be carried out in the mean field setting by changing the type distribution W of the agents, and the resulting system then continues to fulfill the assumptions of the mean field model in Theorem 5.1. We formalize this in the following proposition:

Proposition 5.2 *For each $\alpha \in [0,1]$ and $\theta \in [0,1]$, there exists a corresponding $\theta' \in [0,1]$, such that the reward probability is the same, $\forall f \in [0,1]:\ \alpha Q(\theta, f) = Q(\theta', f)$.*

Proof. First, observe from (5.14) that $Q(0,f) = 0$. As $\alpha < 1$:

$$0 = Q(0,f) \leq \alpha Q(\theta, f) \leq Q(\theta, f). \tag{5.16}$$

Additionally, note from (5.15) that Q is continuous in θ. By the intermediate value theorem, there exists $\theta' \in [0, \theta]$ such that

$$Q(\theta', f) = \alpha Q(\theta, f). \tag{5.17}$$

□

Therefore, we can define a function $g_{\boldsymbol{\alpha}} : [0,1]^n \to [0,1]^n$ that maps θ to θ' and satisfies (5.17). The modified type distribution can then be defined as the pushforward $W_{\boldsymbol{\alpha}} = g_{\boldsymbol{\alpha}\#} W$.

The population profile $\boldsymbol{f}$ now depends on $\boldsymbol{\alpha}$ and we can formulate the minimization problem as follows:

$$\min_{\boldsymbol{\alpha} \in \Delta^n} \frac{1}{2} \| \boldsymbol{f}(\boldsymbol{\alpha}) - \boldsymbol{f}^* \|_2^2, \tag{5.18}$$

where $\Delta^n = \{\alpha \in \mathbb{R}_+^n \,|\, \sum_i \alpha_i = 1\}$ is the n-dimensional unit simplex, a constraint introduced to guarantee a unique solution, as we conjecture $\boldsymbol{f}(\boldsymbol{\alpha})$ to be invariant under multiplication of $\boldsymbol{\alpha}$ by a scaling factor greater than zero. To calculate the stationary profile $\boldsymbol{f}(\boldsymbol{\alpha})$, we resort to a numerical approach. However, to increase efficiency, we make the following observation:

Proposition 5.3 *Consider a multi-agent bandit system $S = (m,n,Q,W,\sigma)$ that has a MFSS $(\boldsymbol{f},\mu)$. Assume that the reward $Q(\theta,f)$ is strictly monotonically decreasing in f, and all agents prefer winning arms, i.e., σ satisfies*

$$\sigma(\boldsymbol{z}+\boldsymbol{w}_j) \geq \sigma(\boldsymbol{z}+\boldsymbol{l}_j), \tag{5.19}$$

for all $j \in \{1,\dots,m\}$, where $\boldsymbol{w_j} = \boldsymbol{e}_{2j-1}$, $\boldsymbol{l}_j = \boldsymbol{e}_{2j}$ are unit vectors corresponding to a win and a loss on arm j, respectively. Fix $i \in \{1,\dots,m\}$, then let $\hat{S} = (m,n,Q,\hat{W},\sigma)$ with $\hat{W} = g_{\boldsymbol{\alpha}\#}W$ and $\alpha_i \in [0,1]$, $\alpha_{j\neq i} = 1$. Then, $\hat{S}$ has a MFSS $(\hat{\boldsymbol{f}},\hat{\mu})$ with $\hat{f}_i \leq f_i$.

Proof. We use Proposition 5.2 to rewrite $\hat{S}$ using the adjusted reward function αQ instead of the adjusted type distribution. For simplicity, we drop the subscript m and the type θ as both systems now have equal W.

The existence of the MFSS $(\hat{\boldsymbol{f}},\hat{\mu})$ follows from Theorem 5.1. We continue the proof by contradiction: Assume $\hat{f}_i > f_i$. Therefore, more agents must pick arm i:

$$\hat{f}_i = \mathbb{E}_{z\sim\hat{\mu}}\left[\sigma(z,i)\right] > \mathbb{E}_{z\sim\mu}\left[\sigma(z,i)\right] = f_i. \tag{5.20}$$

As σ fulfills (5.19), it holds that $\hat{\mu}(Z_{x,i}) > \mu(Z_{x,i})\ \forall x \in [0,1]$ where $Z_{x,i} := \{z : \frac{z_{2i-1}}{z_{2i-1}+z_{2i}} \leq x\}$. This requires more agents winning on arm i. As $\alpha Q < Q$ and Q is strictly monotonically decreasing in f, $\hat{f}_i < f_i$ must hold to increase the win rate on arm i, which contradicts $\hat{f}_i > f$. □

By using UCB as σ and Q as in (5.14), the conditions in Proposition 5.3 are fulfilled. Therefore, if we obtain a population profile $\boldsymbol{f}(\boldsymbol{\alpha})$ using a numerical method, we at least know in which direction to move $\boldsymbol{\alpha}$ to achieve a value closer to the desired profile $\boldsymbol{f}^*$.

Based on the discussion above, we propose to optimize (5.18) using a variant of projected gradient descent with a fixed step size s and a rough estimate of the gradient of α:

$$\nabla\boldsymbol{\alpha}_t \approx \boldsymbol{f}(\boldsymbol{\alpha}_t) - \boldsymbol{f}^*, \quad \boldsymbol{\alpha}_{t+1} = P_{\Delta^n}(\boldsymbol{\alpha}_t - s\cdot\nabla\boldsymbol{\alpha}_t), \tag{5.21}$$

where P_{Δ^n} is the projection on the unit simplex.

Algorithm 5.1 summarizes our proposed method. In Section 5.7, numerical results show that our algorithm quickly converges to a near-optimal solution.

5.7 Numerical Results

In this section, we conduct four experiments to illustrate the convergence of the population profile to the MFSS and show the effectiveness of the proposed load-balancing algorithm. We consider a network with $m \in \{100, 10^4\}$ agents and $n \in \{2,8\}$ arms. Each agent transmits tasks of random size $s_k \in [0.5,1]$ Mbit over a Rayleigh fading channel with parameter 1 (Chen et al., 2019). The transmission power is $p_0 = 200$ mW (Chen et al.,

Algorithm 5.1 Load-balancing algorithm

1: **def** loadBalancing($\boldsymbol{f}^*$, s):
2: **Input** Desired population profile $\boldsymbol{f}^*$ and step size s
3: **Output** Set of parameters $\boldsymbol{\alpha}$ such that $\boldsymbol{f}(\boldsymbol{\alpha}) \approx \boldsymbol{f}^*$
4: $\boldsymbol{\alpha} \leftarrow [1/n, \ldots, 1/n]$
5: **while** not converged **do**
6: $\quad \boldsymbol{f}_t \leftarrow$ simulateMAB($\boldsymbol{\alpha}$)
7: $\quad \nabla\boldsymbol{\alpha} \leftarrow \boldsymbol{f}_t - \boldsymbol{f}^*$
8: $\quad \boldsymbol{\alpha} \leftarrow$ projectSimplex($\boldsymbol{\alpha} - s \cdot \nabla\boldsymbol{\alpha}$)
9: **end while**
10: return $\boldsymbol{\alpha}$

2019), the noise power density is -174 dBm, and $I_{ki} \in [8, 12]$ mW (Xiao et al., 2020). The bandwidth $B = 10$ MHz (Chen et al., 2019) is shared between devices transmitting to the same server. Similarly, the available CPU cycles, $F = 4$ GHz (Chen et al., 2019), are divided among the offloading devices. For $n = 2$, we choose $\boldsymbol{f}^* = \Big[0.2, 0.8\Big]$, for $n = 8$ we choose $\boldsymbol{f}^* = \Big[0.07, 0.08, \ldots, 0.13, 0.3\Big]$. One can select the profile flexibly as long as the values are not extreme.

We select the maximum delay as $d_{\max} = 2m \cdot (0.26)$. This value is the average delay per agent, which we obtained empirically when simulating the communication model, multiplied by $2m$ to accelerate the convergence of our algorithm. Since we require $\boldsymbol{\alpha}$ to sum to 1, the reward of each server reduces. If $d_{\max}$ is too low, the server gives very few rewards in total, and tuning α for this server (further reducing the rewards) does not change the agent choice too much.

To obtain the desired $\boldsymbol{\alpha}$ for the target profile, Algorithm 5.1 is executed for 150 optimization steps. In line 6 of Algorithm 5.1, the MAB game is simulated for 300 steps. To stabilize the optimization, we choose $\boldsymbol{f}_t$ as the average profile over the last 50 steps. This enables obtaining good convergence even for only 100 agents, where the profile is still very noisy in equilibrium.

Fig. 5.1 shows the performance of our optimization procedure for a network with 2 arms, tested with 100 agents in one scenario and 10^4 agents in another. It depicts the fraction of agents choosing arm 1 over 300 simulation steps, comparing the unadjusted setting (blue) against the adjusted setting (orange) obtained after running our optimization algorithm. In the first case, because all fading channels are i.i.d., the devices are evenly distributed across the servers. After adjusting the population profile, the fraction of devices choosing the first server (arm) converges to the target value of $f_1^* = 0.2$.

Fig. 5.2 provides the training curves of Algorithm 5.1. It reports $\|\boldsymbol{f}^* - \boldsymbol{f}_t\|_2^2$, where $\boldsymbol{f}_t$ is the profile obtained by averaging the last 50 steps of a 300-step MAB simulation. We observe good convergence in all four experiments.

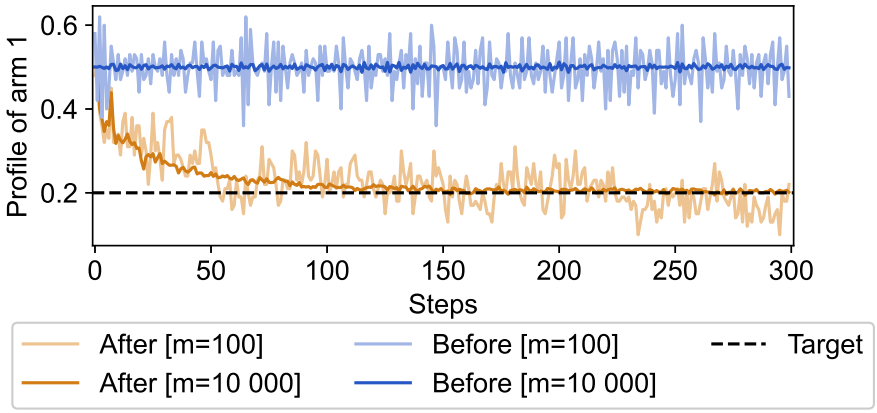


Figure 5.1: The fraction of agents selecting arm 1, f_1, for 100 and 10^4 agents before (blue) and after (orange) applying the load-balancing algorithm. The value of f_1^* is 0.2, and there are 300 optimization steps.

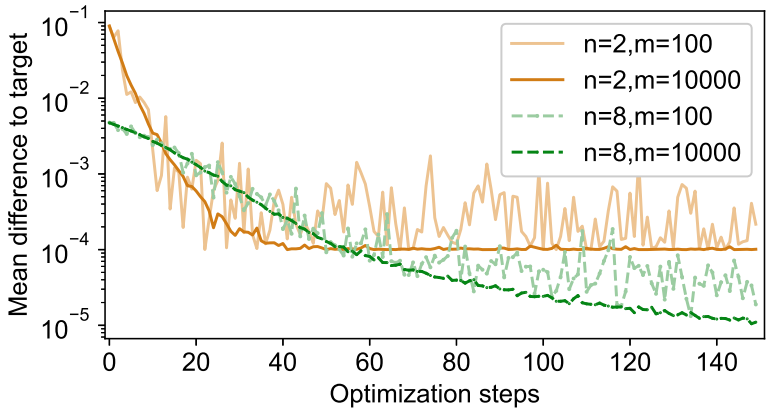


Figure 5.2: The mean distance between f and f^* over 150 optimization steps for networks with two (orange) and eight (green) servers with 100 and 10^4 agents in each case.

5.8 Conclusion

We investigated a decentralized competitive task offloading and load balancing problem under uncertainty about available communication- and computation resources and tasks' size. We modeled it as a mean field MAB game and proved it has a unique MFSS. We proposed a novel decentralized decision-making policy that guarantees the convergence of servers' load to a target population profile according to theoretical- and numerical analysis.

Future work could extend this research in several directions. One of them is to investigate a more complicated system model, for example, the impact of interference on the

system performance and its convergence to the MFSS. Furthermore, one can relax the assumption of a known target population profile and explore designing f^* to minimize offloading costs or ensure fairness across servers. Lastly, we plan to conduct a more in-depth theoretical analysis of the proposed mean field problem. Of particular interest are theoretical guarantees for the convergence of our proposed algorithm, for example, by analyzing the convexity of $f(\alpha)$ and using some methods to directly estimate $f(\alpha)$ from α, which might accelerate the simulation process.

Chapter 6

FL in UAV-Enhanced Networks: Joint Coverage and Convergence Time Optimization

Note: This work was published as a journal paper in *IEEE Transactions on Wireless Communications* (Yahya et al., 2024b). A shorter version of this work was presented at the European Wireless Conference 2022 (Yahya et al., 2022) and is not included in this thesis.

Abstract Federated learning (FL) involves several devices that collaboratively train a shared model without transferring their local data. FL reduces the communication overhead, making it a promising learning method in UAV-enhanced wireless networks with scarce energy resources. Despite the potential, implementing FL in UAV-enhanced networks is challenging, as conventional UAV placement methods that maximize coverage increase the FL delay significantly. Moreover, the uncertainty and lack of a priori information about crucial variables, such as channel quality, exacerbate the problem. In this chapter, we first analyze the statistical characteristics of a UAV-enhanced wireless sensor network (WSN) with energy harvesting. We then develop a model and solution based on the multi-objective multi-armed bandit theory to maximize the network coverage while minimizing the FL delay. Besides, we propose another solution that is particularly useful with large action sets and strict energy constraints at the UAVs. Our proposal uses a scalarized best-arm identification algorithm to find the optimal arms that maximize the ratio of the expected reward to the expected energy cost by sequentially eliminating one or more arms in each round. Then, we derive the upper bound on the error probability of our multi-objective and cost-aware algorithm. Numerical results show the effectiveness of our approach.

6.1 Introduction

The advances in the hardware for on-device intelligence enable unmanned aerial vehicles (UAVs) to offer computation-intensive services based on machine learning methods, such as image classification (Zhang et al., 2022) and smart agriculture (Friha et al., 2021). Implementing centralized machine learning methods in UAV networks is often infeasible due to the requirement of transferring large data volumes and privacy hazards. To mitigate such shortcomings, federated learning (FL) has emerged as an alternative learning paradigm in UAV-enhanced networks.

FL is a distributed learning algorithm that preserves the clients' privacy and reduces communication costs by training a shared model mainly locally in the clients without sharing their raw data. In each global iteration, the server transmits the global model to the clients to update it using their local data. Then, they send the updated local models to the server that aggregates them to start a new iteration. This iterative process of transmitting and updating the models continues until achieving the desired global accuracy (McMahan et al., 2017). FL facilitates various applications and services in UAV-enhanced networks. These include communication, network management, and computation offloading (Nguyen et al., 2021).

One major challenge of FL is the potentially lengthy convergence, which stems from (i) the prolonged transmission time of the local and global updates over the wireless channels, (ii) the computation delays at the FL clients to update the local models, and (iii) the iterative global updates. There are different methods to address this issue, such as client selection (CS) (Chen et al., 2020, Huang et al., 2021b, Xia et al., 2020, Cho et al., 2020, Xia et al., 2021, Xu et al., 2021), resource allocation (Dinh et al., 2021, Zeng et al., 2020), and update compression (Becking et al., 2022). Client selection is one of the most common approaches for reducing the FL convergence time. It selects the clients that lead to faster convergence to participate in FL, which makes it particularly useful in networks with many clients and a few channels. The CS criteria often depend on the client's communication and computation delays in each global iteration (Xia et al., 2021), the clients' local losses (Cho et al., 2020), or their data type (Zhang et al., 2021). To ensure the clients' participation in FL training, (Huang et al., 2021b, Xia et al., 2020, 2021, Cho et al., 2020) tackle the fairness issue in CS. Alternatively, Dinh et al. (2021) use the resource allocation approach to balance two conflicting objectives, minimizing the FL time and reducing the energy consumption. Convergence time minimization is a crucial research topic also in UAV-enhanced networks (Zeng et al., 2020, Yang et al., 2021a). The authors in (Zeng et al., 2020) design a joint power allocation and scheduling scheme for a UAV swarm to optimize the FL convergence while controlling energy consumption. Finally, asynchronous FL schemes shorten the convergence time (Yang et al., 2021a) as the servers do not wait for all local updates.

Another major challenge to optimizing the FL performance is the uncertainty due to the

randomness and lack of information, for example, in channel qualities, resource availability, and network density or topology. Even though the channel state information and the variables above are unavailable in realistic scenarios, the majority of the cutting-edge research assumes otherwise (Dinh et al., 2021, Zeng et al., 2020, Yang et al., 2021a). Still, a few works use machine learning-based methods to overcome this challenge. The most relevant example to our work-example is the multi-armed bandit (MAB) framework, where the network manager learns the network's statistical information sequentially during the FL training process (Xia et al., 2020, Huang et al., 2021b, Cho et al., 2020, Xia et al., 2021, Xu et al., 2021, Cao et al., 2021). More details on the MAB framework are given in Section 6.1.2. Table 6.1 summarizes the most relevant state-of-the-research.

Paper	**MAB**	**UAV**	**Highlights**	**Other considerations**
Client selection				
Xia et al. (2020)	✓	✗	Minimizes the transmission and local computation time for two cases, clients with i.i.d and balanced datasets and non-i.i.d. and unbalanced datasets for which a fairness constraint is added.	Fairness in CS and the clients' availability.
Huang et al. (2021b)	✓	✗	Uses contextual combinatorial bandits (C^2MAB) to estimate the model exchange time between the clients and the server and balances between the fairness in CS and the FL training efficiency.	Fairness in CS and the clients' availability.
Cho et al. (2020)	✓	✗	Selects the clients with higher local losses for faster FL convergence using MABs and increases the fairness in user selection.	Fairness in CS.
Xia et al. (2021)	✓	✓	Reduces the number of communication rounds by selecting the clients based on their update staleness and weight divergence, then it shortens the average time interval per round by performing latency-based client scheduling using MABs.	Fairness in CS.

Xu et al. (2021)	✓	✗	Uses the ε-greedy algorithm to select the clients that jointly minimize the number of global rounds (clients with significant local losses) and the delay per round (clients with lower latency).	–
Yang et al. (2021a)	✗	✓	Proposes an asynchronous FL algorithm for faster convergence. The UAV servers employ an algorithm based on asynchronous advantage actor-critic for joint device selection, UAVs placement, and resource management.	The reward function captures the FL time and accuracy loss.
Resource Allocation				
Zeng et al. (2020)	✗	✓	Minimizes the number of global rounds by jointly designing the power allocation and scheduling in a UAV network. This work considers the transmission and UAV flying power. The optimization problem is solved using the sample average approximation and dual method.	The UAVs' stability and their flying and transmission power consumption
Dinh et al. (2021)	✗	✗	Proposes an FL algorithm named FEDL for heterogeneous clients and characterizes the trade-off between the local and global iterations. It also proposes a resource allocation problem in wireless networks to achieve a trade-off between the FL time and the energy consumption.	The heterogeneity in the clients datasize and CPU frequency.
UAV coverage				
Yahya and Maghsudi (2022)	✗	✓	Minimizes the FL delay by optimizing the locations of the UAVs to achieve homogeneous coverage across the UAVs to minimize the maximum computation delay, followed by a bandwidth and power allocation step to minimize the communication delay.	The power is allocated to achieve max-min fairness in the data rate.

Ours	✓	✓	Proposes a MO-MABs-based approach to jointly maximize the UAVs coverage and minimize the FL local model update time under. It proposes another solution that is particularly useful when there is a large number of arms. It is based on a scalarized best arm identification algorithm.	Uncertainty about the channel and the sensors' location and availability.

Table 6.1: A comparison between the FL convergence time minimization approaches.

Thanks to their flexibility and low cost, UAVs integrate well into 5G, 6G, and internet-of-things (IoT) networks as aerial base stations to improve their performance and enable new applications (Friha et al., 2021, Zeng et al., 2020, Yahya and Maghsudi, 2022, Yang et al., 2021a,b,c, Peng and Wang, 2023). For example, UAVs are used in private content caching while ensuring age-of-information aware, fair, and energy-efficient content delivery (Yang et al., 2021b). In such applications, ML methods are widely applicable. For example, deep neural networks predict the mobile users' locations to perform UAV-network slicing to guarantee coverage and the ultra-reliable low-latency communications (URLLC) requirements of control signals (Yang et al., 2021c). Moreover, using deep reinforcement learning, one can optimize the phase shifts of reconfigurable intelligent surfaces for energy harvesting (Peng and Wang, 2023) and the energy efficiency of blockchain operations in UAV-assisted computation offloading in IoT networks (Lan et al., 2023). In this chapter, we optimize the performance of a FL UAV-enhanced network under high uncertainty and conflicting objectives using MABs.

6.1.1 Motivation

As described above, the UAVs might engage in an FL procedure, each using the data collected locally in its coverage area (Brik et al., 2020), one significant example appears in IoT for agriculture for crop monitoring, fertilizer management, and data collection (Friha et al., 2021). In such a scenario, the UAVs' placement and coverage substantially affect the FL convergence time, an issue that remains unaddressed to a great extent. The conventional approaches of UAV placement for coverage maximization prolong the FL convergence delay drastically due to the inhomogeneous sensor- or user-density in the UAVs' coverage area. Larger data volumes often require more processing time, so the FL server waits longer for slower UAVs that cover dense locations to update the global model. Such a straggler effect increases the overall delay. In Yahya and Maghsudi (2022), we proposed an approach to jointly maximize the network coverage and minimize the FL convergence time by effective UAVs' placement and radio resource management. The UAVs' placement ensured higher homogeneity in the number of sensors covered by each

UAV, thus, similar processing times.
This chapter proposes a novel method to jointly maximize the sensor coverage and minimize the FL convergence time in a UAV-enhanced network under uncertainty about the sensors' distribution and channels' quality. It also considers the non-deterministic sensors' activity due to opportunistic energy harvesting. The two objectives are conflicting since increasing the sensor coverage results in heavier computation and extended delays. To solve this problem under uncertainty, we use a multi-objective multi-armed bandits (MO-MAB) framework (Drugan and Nowe, 2013) merged as an alternative learning paradigm in UAV networks. With this approach, we provide a solution for a realistic UAV network that is robust to information shortage.

6.1.2 MO-MAB as a Solution Framework

MAB is an online sequential decision-making problem where, in each round, an agent selects an arm from a set to receive a reward sampled from its apriori unknown reward-generating process. The agent intends to choose the arms to minimize its long-term cumulative regret (Maghsudi and Hossain, 2016). In the multi-objective version of the MAB problem, each arm returns a reward vector, where each element is the reward of the corresponding objective (Drugan and Nowe, 2013). Since there are multiple objectives, there can be several optimal reward vectors; thus, several optimal arms (Drugan and Nowe, 2013), (Rădulescu et al., 2020). Hence, the agent aims to simultaneously minimize the regret in all objectives by playing optimal arms fairly.
Drugan and Nowe proposed two ways to obtain the set of optimal arms in a MO-MAB problem: the Pareto partial order approach and the scalarization approach (Drugan and Nowe, 2013). The former maximizes the reward vectors directly in the multi-objective reward space by looking at the dominance relationships between arms, while the latter uses a set of appropriate scalarization functions to transform the multi-objective problem into a single-objective one and find the optimal arms. In this chapter, we solve our bi-objective problem of coverage maximization and convergence time minimization using both approaches and compare their performance.
One challenge in our problem formulation is the large number of arms. In addition, the scalarized multi-objective approach requires finding the optimal arm for each scalarization function. To address these issues, we develop an algorithmic solution to our bi-objective optimization problem based on the best-arm identification (BAI) algorithm (Audibert and Bubeck, 2010). The fixed-budget BAI algorithm aims at finding the optimal arm with high probability in a fixed number of rounds (Audibert and Bubeck, 2010, Shahrampour et al., 2017). Consequently, the algorithm's performance does not concern the cumulative regret but the quality of the final recommendation, i.e., the probability that the recommended arm is optimal.
In each round, the *general sequential elimination algorithm* (Shahrampour et al., 2017)

eliminates one (Audibert and Bubeck, 2010, Drugan and Nowé, 2014) or more arms with the lowest average rewards, allowing the algorithm to scale well with the increasing number of arms (see Table 1 in Shahrampour et al. (2017)). Our proposed solution extends that algorithm to the multi-objective scenario with one crucial adjustment: the optimal arm is defined as the one that maximizes the ratio of the expected reward obtained from selecting an arm to the incurred cost. This cost depends on the network coverage and transmission delays and accounts for the random energy consumed by the UAVs in updating and transmitting the local models. Therefore, the proposed algorithm differs from the approach in Xia et al. (2016) that eliminates one arm per round assuming that the reward and cost are independent. Then, we find an upper bound on the error probability for our proposed algorithm. The fixed budget setting of the proposed algorithm allows us to estimate the required energy of the solution.

6.1.3 Our Contributions

This chapter considers a UAV-enhanced WSN in which the UAVs collect sensor data and engage in FL with a ground server. It addresses the problem of UAV placement, aiming to simultaneously maximize the network coverage and minimize the FL convergence delay under high uncertainty about the sensors' placement, activity, and channel conditions. As these two objectives are conflicting, this chapter proposes solutions based on bi-objective-MABs (BO-MABs). The following list outlines the key contributions.

- Analyzing the components of the FL time and obtaining their statistical distributions.
- Formulating the problem as an optimization problem with two conflicting objectives: maximizing the sensor coverage, and minimizing the total FL delay.
- Modelling and solving the problem in a BO-MAB framework using both the Pareto partial ordering and scalarization. Then, the simulation results for the two methods are presented.
- Given the possibility of having many arms and the limited energy resources at the UAVs, the chapter proposes a scalarization-based sequential elimination algorithm for BAI for the BO-MAB scheme. In this algorithm, the best arm is defined as the one that maximizes the ratio of the expected reward to the expected energy cost for an energy-efficient solution. Afterwards, an upper bound to the error probability is derived for the proposed algorithm.
- The proposed algorithm is used to find the UAV configuration that achieves the best coverage and convergence delay per energy cost. The numerical results show the algorithm's performance for an FL network training the MNIST dataset for image classification.

6.1.4 Chapter Organization

Section 6.2 presents the system model. Section 6.3 analyzes the FL convergence time and shows the dependency of the convergence time on the network coverage. In Section 6.4, the problem is formulated as a bi-objective optimization problem. Then, in Section 6.5, the multi-objective MAB problem is described and then is used in Section 6.6 to solve the joint problem of coverage maximization and convergence time minimization. In Section 6.7, a computationally efficient solution is proposed using a scalarization-based sequential elimination algorithm for best arm identification. Section 6.8 presents the numerical results and Section 6.9 concludes.

6.2 System Model

The system model is illustrated in Fig. 6.1. In this WSN, the sensors are located randomly in a square region of edge length E. The sensors' locations and density distribution are unknown to the UAVs and the ground server. All the sensors are homogeneous in their transmit power and data size, and each sensor relies on ambient energy harvesting to operate. The network also includes a set $\mathcal{U}$ of U rotary-wing UAVs with identical computation capabilities. Each sensor collects data and transmits it to a UAV; thus, a sensor communicates with a UAV if (i) it is within that UAV's coverage region and (ii) it has sufficient energy. Due to the random nature of energy harvesting, the number of transmitting sensors (active) at a given time is random. Each UAV collects data from the active sensors in its coverage region and engages in the FL process with the ground server over a dedicated communication link. This system was motivated by many applications such as the IoT for agriculture and natural disaster analysis. The proposed framework is easily adaptable to other system models as long as network delay and coverage remain uncertain. The first arises due to the randomness in the communication delays and the UAVs' processing capabilities, whereas the latter arises by the randomness in the number of sensors or their availability.

6.2.1 Federated Learning in the UAV-Enhanced Network

Each UAV $i \in \mathcal{U}$ collects D_i data samples $\{\boldsymbol{x}_{il}, y_{il}\}_{l=1}^{D_i}$ from the sensors in its coverage region, $\boldsymbol{x}_{il} \in \mathbb{R}^d$ is a d-feature input vector and $y_{il} \in \mathbb{R}$ is the corresponding label. Then, each UAV trains and updates the local model $\boldsymbol{\omega}$ using its D_i data samples. We use $f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il})$ to denote the loss function for the lth data sample at UAV i. The average loss function of UAV i, $F_i(\boldsymbol{\omega}, \boldsymbol{x}_{i1}, y_{i1}, \ldots, \boldsymbol{x}_{iD_i}, y_{iD_i})$, or simply $F_i(\boldsymbol{\omega})$, is

$$F_i(\boldsymbol{\omega}) \coloneqq \frac{1}{D_i} \sum_{l=1}^{D_i} f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il}). \tag{6.1}$$

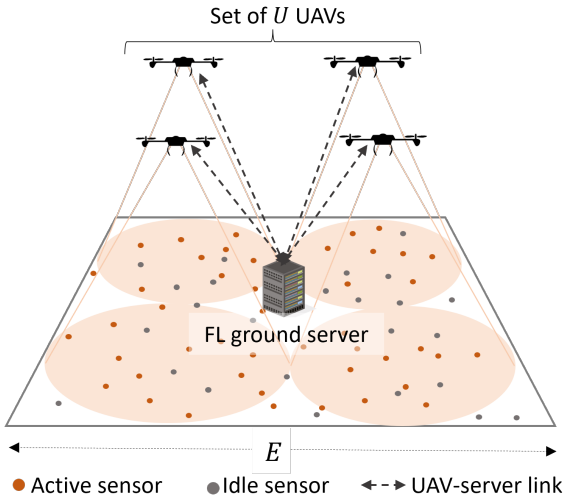


Figure 6.1: A UAV-enhanced sensor network. Each UAV collects data from the active sensors in its coverage region and performs FL with the ground sever.

Let $D = \sum_{i=1}^{U} D_i$. The global loss function is defined as (Dinh et al., 2021)

$$F(\boldsymbol{\omega}) := \sum_{i=1}^{U} \frac{D_i}{D} F_i(\boldsymbol{\omega}) = \frac{1}{D} \sum_{i=1}^{U} \sum_{l=1}^{D_i} f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il}), \tag{6.2}$$

The goal of FL is to find a parameter $\boldsymbol{\omega}^* \in \mathbb{R}^d$ that minimizes the global loss function

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^d} F(\boldsymbol{\omega}). \tag{6.3}$$

Without loss of generality, the *FEDL* FL algorithm (Dinh et al., 2021) is used in this work to solve the problem in (6.3). The algorithm assumes that $F_i(\boldsymbol{\omega})$ is L-smooth and β-strongly convex, which holds for several applications, such as the l_2-regularized linear regression model with $f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il}) = \frac{1}{2}\left(\langle \boldsymbol{x}_{i,l}, \boldsymbol{\omega} \rangle - y_{i,l}\right)^2 + \frac{\beta}{2}||\boldsymbol{\omega}||^2$, where $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ is the inner product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ and $||\boldsymbol{v}|| = \langle \boldsymbol{v}, \boldsymbol{v} \rangle$ is the 2-norm of $\boldsymbol{v}$.
In brief, each global iteration m of FEDL starts when the ground server transmits a model to the UAVs. Then, each UAV i updates the model using the D_i local data samples collected from the active sensors in its coverage region with a predetermined accuracy $\kappa \in (0,1)$. The local update time at UAV i is

$$T_i^{\text{cp}} = I(\kappa) \frac{\delta D_i}{f_{\text{CPU},i}}, \tag{6.4}$$

where $I(\kappa)$ is the number of local iterations required to achieve local accuracy κ, δ (cycle/sample) is the number of computation cycles per data sample, and f_{CPU} is the UAV's CPU frequency. Afterward, each UAV sends its current model to the server to update the global model for the next round. The global iterations continue until the stopping condition is met. By definition, the FL converges when the difference between the current and the minimal loss functions is no more than the global accuracy ε, i.e., $F(\boldsymbol{\omega}^m) - F(\boldsymbol{\omega}^*) \leq \varepsilon$. However, since $F(\boldsymbol{\omega}^*)$ is unknown to the server, the algorithm stops after a predetermined number of global iterations.

Algorithm 6.1 The FEDL Algorithm (Dinh et al., 2021)

1: In the first global iteration $m = 0$, the server generates and broadcasts an initial global model $\boldsymbol{\omega}^0$ to all UAVs. It also determines the local accuracy parameter $\kappa \in (0,1)$.
2: In the following global iterations, each UAV i receives $\boldsymbol{\omega}^{m-1}$ and $\nabla \bar{F}^{m-1}$ from the server. It then solves $\boldsymbol{\omega}_i^m = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^d} J_i^m(\boldsymbol{\omega})$,

$$J_i^m(\boldsymbol{\omega}) := F_i(\boldsymbol{\omega}) + \langle \eta \nabla \bar{F}^{m-1} - \nabla F_i(\boldsymbol{\omega}^{m-1}), \boldsymbol{\omega} \rangle, \tag{6.5}$$

where η is the learning rate. The resulting $\boldsymbol{\omega}_i^m$ satisfies the accuracy $\kappa \in (0,1)$, i.e.,

$$||\nabla J_i^m(\boldsymbol{\omega}_i^m)|| \leq \kappa ||\nabla J_i^m(\boldsymbol{\omega}^{m-1})||, \quad \forall i \in \mathcal{U}, \tag{6.6}$$

$\kappa = 0$ solves the local problem optimally, whereas $\kappa = 1$ means that the local model doesn't change.
3: The UAVs send their local models $\boldsymbol{\omega}_i^m$ and local gradients $\nabla F_i(\boldsymbol{\omega}_i^m)$ to the server. It collects them to calculate $\boldsymbol{\omega}^m = \sum_{i=1}^U \frac{D_i}{D} \boldsymbol{\omega}_i^m$ and $\nabla \bar{F}^m = \sum_{i=1}^U \frac{D_i}{D} \nabla \bar{F}_i(\omega_i^m)$. The server then transmits these values to all UAVs.
4: The global iterations, i.e., the collection of updating and transmission processes repeats until convergence, i.e., until achieving a certain difference between the current and the minimal loss function. Formally, $F(\boldsymbol{\omega}^m) - F(\boldsymbol{\omega}^*) \leq \varepsilon$.

6.2.2 Transmission Probability of Energy Harvesting Sensors

The wireless sensors obtain their required energy via ambient energy harvesting, a convenient energy resource in IoT in agriculture (Friha et al., 2021). The sensors use a harvest-use scheme. A sensor transmits its data to a UAV only if it lies in its coverage region and has sufficient transmission energy. And since ambient energy harvesting is non-deterministic, the sensors' activity is random

We model the energy arrival process at a sensor m as a Poisson process with the rate λ_m. The amount of energy harvested in each arrival v, $Z_{m,v}$, follows an exponential distribution with parameter $\eta_{m,v}$, $Z_{m,v} \sim \text{EXP}(\eta_{m,v})$ (Maghsudi and Hossain, 2017). The energy harvesting phase continues until the k_m energy arrival. Afterward, the sensor transmits its

data to its corresponding UAV. For known $\eta_{m,v}$, one can estimate the number of arrivals; otherwise, it can be selected randomly (Maghsudi and Hossain, 2017). The total amount of energy at sensor m, denoted by Y_m, is the sum of k_m energy arrivals, i.e., $Y_m = \sum_{v=1}^{k_m} Z_{m,v}$. To simplify the analysis, we consider the case of independent and identically distributed (i.i.d) energy arrivals [1]. Thus, $\eta_{m,v} = \eta_m$ for $v = \{1, \ldots, k_m\}$ and the total energy is the sum of k_m i.i.d exponential random variables. As a result, Y_m follows the Erlang distribution with parameters k_m and η_m, i.e., $Y_m \sim \text{Erl}(k_m, \eta_m)$

$$f_{Y_m}(y_m) = \frac{{\eta_m}^{k_m}}{(k_m - 1)!} y_m^{(k_m - 1)} e^{-\eta_m y_m}. \tag{6.7}$$

Let E_T be the amount of energy required for a sensor m to transmit to some UAV. Then, the probability that a covered sensor transmits to the corresponding UAV is

$$\mathbb{P}[Y_m \geq E_\text{T}] = \sum_{n=0}^{k_m - 1} \frac{1}{n!} e^{-\eta_m E_\text{T}} (\eta_m E_\text{T})^n. \tag{6.8}$$

6.2.3 Network Coverage

The network coverage can be maximized by finding the optimal locations of the UAVs in the 3D space and then relocating the UAVs accordingly. However, this approach is computationally costly and power inefficient. In addition, given no information about the sensor's deployment density, it is infeasible to find the optimal UAV placement that maximizes the sensor coverage. To overcome these problems, the UAVs are deployed according to the circle packing theorem in the 2D space and retain a fixed altitude H.[2] Then, the coverage of the UAVs is controlled by adjusting the beamwidth of each UAV while maintaining its location. Details follow.

Each UAV i has a directional antenna with an adjustable beamwidth. Similar to He et al. (2017), the azimuth and elevation half-power beamwidths of the antennas are assumed to be the same and equal to $\theta_i \in (0, \frac{\pi}{2})$ in radians. The approximate antenna gain is (Balanis, 2015)

$$G_i(\theta_i) = \begin{cases} G_0/\theta_i^2, & -\theta_i \leq \varphi \leq \theta_i \\ g_0, & \text{else,} \end{cases} \tag{6.9}$$

where $G_0 = 2.2846$ (He et al., 2017), and g_0 is the gain in the side lobes. In practice, we have $0 < g_0 \ll G_0/\theta_i^2$. For simplicity, in what follows, we assume $g_0 = 0$ (He et al., 2017,

[1] The extension to the case of independent and non-identically distributed (i.ni.d) energy arrivals is straight forward as it only affects the probability of transmission. The distribution of Y_m for this case is given in Maghsudi and Hossain (2017) along with the conditions required for approximating the distribution of Y_m by a Normal or an Erlang distributions.

[2] Our work applies to the case where the sensor's distribution is known, but their availability is random. In this case, other initial UAV placements might suit better than circle packing.

Yang et al., 2019b). In the rest of the chapter, the subscript θ_i is added to the variables that depend on the beamwidth.
The radius of the coverage disk formed by the main lobe of the antenna of UAV i is $R_{i,\theta_i} = H\tan(\theta_i)$. The path loss of the link between UAV i and a sensor located at the edge of the coverage region R_{i,θ_i} in dB yields (Al-Hourani et al., 2014)

$$L_{i,\theta_i}^{\mathrm{dB}} = \frac{A}{1+a\exp\left(-b[\frac{180}{\pi}\tan^{-1}\left(\frac{H}{R_{i,\theta_i}}\right)-a]\right)} + 10\log\left(R_{i,\theta_i}^2+H^2\right)+Y, \tag{6.10}$$

where $A = \eta_{\mathrm{LoS}} - \eta_{\mathrm{NLoS}}$ is the mean excessive loss in the line-of-sight (LoS) and non-LOS (NLoS) paths, while a and b are constants that depend on the environment (Al-Hourani et al., 2014). Moreover, $Y = 20\log\left(\frac{4\pi f_c}{c}\right)+\eta_{\mathrm{NLoS}}$, where f_c is the carrier frequency and c is the speed of light.
The maximum beamwidth $\theta_{i,\max}$ of UAV i is determined by the minimum acceptable receive power $P_{\mathrm{r,min}}$ at UAV i defined to be

$$P_{\mathrm{r,min}} \coloneqq P_{\mathrm{t}} + G_{i,\theta_i} - L_{i,\theta_{i,\max}}^{dB}, \tag{6.11}$$

where P_{t} is the power transmitted by a wireless sensor and it is equal for all sensors. G_{i,θ_i} is the antenna gain of UAV i when its beamwidth is θ_i. Given $\theta_{i,\max}$, one can determine the maximum coverage area $R_{i,\theta_{i,\max}}$ of UAV i.
The probability distribution of the number of sensors is unknown to the server. However, we assume this distribution belongs to the class of Poisson distributions parameterized by an unknown rate λ (sensor/m^2). The probability distribution of the number of sensors located in the coverage disk of UAV i of area $\Omega_{i,\theta_i} = \pi R_{i,\theta_i}^2$ is $\mathbb{P}[N_{i,\theta_i}^{\mathrm{total}} = k] = \frac{(\lambda\Omega_{i,\theta_i})^k e^{-\lambda\Omega_{i,\theta_i}}}{k!}$.
Now since (6.8) gives the probability that a sensor has sufficient harvested energy and it is independent of $\mathbb{P}[N_{i,\theta_i}^{\mathrm{total}} = k]$, the number of active sensors N_{i,θ_i} is $\mathbb{P}[N_{i,\theta_i} = k] = \mathbb{P}[N_{i,\theta_i}^{\mathrm{total}} = k]\mathbb{P}[Y_m \geq E_{\mathrm{T}}]$. Hence, we have for $k = 0, 1, \ldots$

$$\mathbb{P}[N_{i,\theta_i} = k] = \frac{(\lambda\Omega_{i,\theta_i})^k e^{-\lambda\Omega_{i,\theta_i}}}{k!} \sum_{n=0}^{k_m-1} \frac{1}{n!} e^{-\eta_m E_{\mathrm{T}}} (\eta_m E_{\mathrm{T}})^n, \tag{6.12}$$

The average number of active sensors covered by UAV i is

$$\bar{N}_{i,\theta_i} = \lambda\Omega_{i,\theta_i} \sum_{n=0}^{k_m-1} \frac{1}{n!} e^{-\eta_m E_{\mathrm{T}}} (\eta_m E_{\mathrm{T}})^n. \tag{6.13}$$

6.3 The FL Convergence Time and Its Dependency on Coverage

In the proposed UAV-enhanced network, the FL convergence time has three main components:

- The local model update time at the UAVs (computation time);
- The data transmission time between the sensors and the UAVs;
- The local- and global model exchange time between the UAVs and the server.

Changing a UAV's beamwidth influences the number of its covered sensors, hence the number of its collected data samples and its model update time. In addition, the communication time over the sensor-UAV link depends on the beamwidth because it affects the distance between the sensor and the UAV and so the channel quality. More precisely, larger beamwidths worsen the path loss, reduce the antenna gain, and prolong the communication delays. In contrast, the communication delay between each UAV and the server depends on the length and quality of the UAV-server communication link.

6.3.1 Computation Time

The beamwidth of UAV i, θ_i, determines the number of active sensors it covers and with it, the number of data samples D_{i,θ_i} it processes. Given that the sensors sample at the same frequency, i.e., there are D samples per sensor, the average computation time of UAV i yields

$$T^{\text{cp}}_{i,\theta_i} = I(\kappa)\frac{\delta D_{i,\theta_i}}{f_{\text{CPU}}} = I(\kappa)\frac{\delta D\bar{N}_{i,\theta_i}}{f_{\text{CPU}}}, \tag{6.14}$$

where $\bar{N}_{i,\theta_i}$ is the average number of active nodes given by (6.13). That shows that a large beamwidth can result in a high computation time by increasing the coverage. To simplify the derivation of the delay probability distribution, the UAVs' CPU frequency f_{CPU} is assumed to be constant. The amount of energy consumed by UAV i in updating its local model is

$$E^{\text{cp}}_{i,\theta_i} = \frac{\alpha}{2} I(\kappa)\delta D_{i,\theta_i} f^2_{\text{CPU}}, \tag{6.15}$$

where α is the energy capacitance coefficient of the UAVs.
The following proposition gives the probability distribution function of the computation time at UAV i for the Poisson nodes' distribution. The value of $T^{\text{cp}}_{i,\theta_i}$ in (6.14) is an integer multiple τ of $I(\kappa)\frac{\delta D}{f_{\text{CPU}}}$. The mean computation time at the UAV per unit of coverage area is $\nu = I(\kappa)\frac{\delta D\lambda}{f_{\text{CPU}}}$ second/m^2, where λ is the sensor density. The probability distribution

function of the computation time at UAV i, $T_{i,\theta_i}^{\mathrm{cp}}$ is

$$\mathbb{P}\left[T_{i,\theta_i}^{\mathrm{cp}} = \tau\left(I(\kappa)\frac{\delta D}{f_{\mathrm{CPU}}}\right)\right] = \frac{(\nu\Omega_{i,\theta_i})^{\tau}e^{-(\nu\Omega_{i,\theta_i})}}{\tau!}\sum_{n=0}^{k_m-1}\frac{1}{n!}e^{-\eta_m E_{\mathrm{T}}}(\eta_m E_{\mathrm{T}})^n, \quad \tau = 0,1,2,\ldots \tag{6.16}$$

6.3.2 The Communication Time over the UAV-Server Link

As long as a UAV does not move, its link to the ground server and the average delay of the server-UAV communication remains fixed. However, because of the random channel quality, the instantaneous delay can vary. The path loss between UAV i and the ground server, denoted $L_{i,\mathrm{G}}^{\mathrm{dB}}$ is obtainable by replacing R_{i,θ_i} in (6.10) with $d_{i,\mathrm{G}}$ which represents the horizontal distance between the projection of UAV i on the ground and the ground server. The data rate then yields

$$r_{i,\mathrm{G}} = B_{\mathrm{G}}\log\left(1+\frac{P_{\mathrm{t,G}}|g_{i,\mathrm{G}}|^2 G_{i,\mathrm{G}}}{L_{i,\mathrm{G}}\,\sigma_{\mathrm{G}}^2}\right), \tag{6.17}$$

where B_{G} is the bandwidth and $P_{\mathrm{t,G}}$ is the UAV's transmission power. Moreover, $g_{i,\mathrm{G}}$ and $G_{i,\mathrm{G}}$ respectively denote the gain of the Ricean channel and the antenna gain in the direction of the ground server. Finally, σ_{G}^2 is the noise variance. For the downlink and uplink, we assume equal transmission bandwidths and channel reciprocity. Therefore, the transmission time of the global- and local updates with size S_{G} and S_{L} bits, respectively, is given by

$$T_{i,\mathrm{G}} = \frac{S_{\mathrm{G}}}{r_{i,\mathrm{G}}}, \quad T_{i,\mathrm{L}} = \frac{S_{\mathrm{L}}}{r_{i,\mathrm{G}}}. \tag{6.18}$$

The following proposition gives the statistical distribution of the communication time over the UAV-server link for UAV i.

Proposition 6.1 *$T_{i,G}$ is a random variable denoting the communication time of S_M bits over the UAV-server link, its probability distribution is given by*

$$f_{T_{i,G}}(t_{i,G}) = \frac{S_M}{2B_G\gamma_{i,G}\xi_G^2 t_{i,G}^2}\exp\left(\frac{k_{i,G}^2\gamma_{i,G}+e^{\left(\frac{S_M}{B_G t_{i,G}}\right)}-1}{2\gamma_{i,G}\xi_G^2}\right)I_0\left(\frac{k_{i,G}\sqrt{e^{\frac{S_M}{B_G t_{i,G}}}-1}}{\xi_G^2\sqrt{\gamma_{i,G}}}\right), \tag{6.19}$$

where $\gamma_{i,G} = \frac{P_{t,G}G_{i,G}}{L_{i,G}\sigma_G^2}$. S_M stands for either the global update S_G or the local update S_L. $k_{i,G}$ and ξ_G represent the parameters of the χ^2 distribution.

Proof. See Appendix 6.A □

6.3.3 The Communication Time over the Sensor-UAV Link

Similar to the computation time, the UAVs' beamwidth affects the communication time over the sensor-UAV link. By increasing the beamwidth, the antenna gain reduces, while the coverage area becomes bigger, implying a potentially longer UAV-sensor distance. Here, the focus is on the delay experienced by the farthest sensor from the projection of the UAV on the ground because it has the highest path loss. Due to the uncertainty in the sensors' locations, **Proposition 6.2** obtains the expected value of the maximum distance between the UAV and the sensor using the order statistics. Later this value is used in finding the average maximum delay.

Proposition 6.2 *The average maximum distance between UAV i and a sensor located in its coverage region is*

$$\bar{Z}_{i,\theta_i} = \int_0^\infty z_i \lambda \bar{N}_{i,\theta_i} \Omega_{i,\theta_i} e^{(\lambda \Omega_{i,\theta_i}) z_i} \left(1 - e^{(\lambda \Omega_{i,\theta_i}) z_i}\right)^{\bar{N}_{i,\theta_i}-1} dz_i \tag{6.20}$$

Proof. See Appendix 6.B □

The farthest sensor F from UAV i within its coverage disk experiences the highest path loss denoted by $L^{\mathrm{dB}}_{\bar{Z}_{i,\theta_i}}$, its value is obtainable from (6.10) by replacing R_{i,θ_i} with $\bar{Z}_{i,\theta_i}$. The sensor's transmission rate, $r_{\bar{Z}_{i,\theta_i}}$, yields

$$r_{\bar{Z}_{i,\theta_i}} = B_F \log_2 \left(1 + \frac{P_{\mathrm{t}} |g_F|^2 G_{i,\theta_i}}{L_{\bar{Z}_{i,\theta_i}} \sigma^2_{\mathrm{F}}}\right), \tag{6.21}$$

where B_{F} is the channel bandwidth and P_{t} is the sensor's transmit power. In addition, g_{F} denotes the gain of the Ricean channel at the farthest sensor and G_{i,θ_i} is the antenna gain of UAV i. σ^2_{F} is the noise variance. Here, the communication between the sensors and the UAVs is over orthogonal channels that mitigate interference, but this work can be extended to other scenarios with interfering sensors by replacing (6.21) with the expression $r_{\bar{Z}_{i,\theta_i}} = B_F \log_2 \left(1 + \frac{P_{\mathrm{t}} |g_F|^2 G_{i,\theta_i} L^{-1}_{\bar{Z}_{i,\theta_i}}}{I_\Psi + \sigma^2_{\mathrm{F}}}\right)$, where Ψ is the set of active sensors interfering with sensor F, and I_Ψ is the total interference power at sensor F. Interference reduces the data rate and results in longer transmission delays. The transmission time of S_{data} bits then yields

$$T^{\mathrm{cm}}_{\bar{Z}_{i,\theta_i}} = \frac{S_{\mathrm{data}}}{r_{\bar{Z}_{i,\theta_i}}}. \tag{6.22}$$

Proposition 6.3 *The probability distribution of the maximum delay over the sensor-UAV*

link assuming no overlap between the UAVs' coverage regions is

$$f_{T_{\bar{Z}_{i,\theta_i}}}(t_{\bar{Z}_{i,\theta_i}}) = \frac{S_{data}}{2B_F\alpha_{\bar{Z}_{i,\theta_i}}\xi_F^2 t_{\bar{Z}_{i,\theta_i}}^2} \times \exp\left(\frac{k_F^2\alpha_{\bar{Z}_{i,\theta_i}} + e^{\frac{S_{data}}{B_F t_{\bar{Z}_{i,\theta_i}}}} - 1}{2\alpha_{\bar{Z}_{i,\theta_i}}\xi_F^2}\right) I_0\left(\frac{k_F\sqrt{e^{\frac{S_{data}}{B_F t_{\bar{Z}_{i,\theta_i}}}} - 1}}{\xi_F^2\sqrt{\alpha_{\bar{Z}_{i,\theta_i}}}}\right) \tag{6.23}$$

where $\alpha_{\bar{Z}_{i,\theta_i}} = \frac{P_t G_{i,\theta_i}}{L_{\bar{Z}_{i,\theta_i}}\sigma_F^2}$. ξ_F *and* k_F *are the parameters of the* χ^2 *distribution.*

Proof. The proof of this proposition is similar to the proof of the distribution of the transmission time over the UAV-server link given by **Proposition 6.1** because the channel has a Ricean distribution in both cases. □

6.3.4 The Total FL Convergence Time

The overall communication time is

$$T_{i,\theta_i}^{\text{cm}} = T_{\bar{Z}_{i,\theta_i}} + T_{i,\text{G}} + T_{i,\text{L}}. \tag{6.24}$$

To find the distribution of the communication time, the simplifying assumption of equal sized local and global updates is made to have equal transmission times. A generalization to the uneven case is straightforward. Since the transmission time over the sensor-UAV link $t_{\bar{Z}_{i,\theta_i}} > 0$ is independent from the transmission time over the UAV-server link, $t_{i,\text{G}} > 0$ the probability distribution of the total communication time $t_{\bar{Z}_{i,\theta_i}}$ is given by their convolution (Papoulis and Pillai, 2002)

$$f_{T_{i,\theta_i}^{\text{cm}}}(t_{i,\theta_i}^{\text{cm}}) = \int_0^{t_{i,\theta_i}^{\text{cm}}} f_{T_{\bar{Z}_{i,\theta_i}}}(t_{\bar{Z}_{i,\theta_i}}) f_{T_{i,\text{G}}}(t_{i,\theta_i}^{\text{cm}} - t_{\bar{Z}_{i,\theta_i}}) dt_{\bar{Z}_{i,\theta_i}}. \tag{6.25}$$

The total convergence time is

$$T_{i,\theta_i}^{\text{total}} = T_{i,\theta_i}^{\text{cp}} + T_{i,\theta_i}^{\text{cm}}. \tag{6.26}$$

Finally, **Proposition 6.4** gives the cumulative distribution function of the total delay.

Proposition 6.4 *The cumulative distribution function of the total delay is*

$$F_{T_{i,\theta_i}^{total}}\left(t_{i,\theta_i}^{total}\right) = \sum_{\tau\in\bar{N}_{i,\theta_i}} \mathbb{P}\left[t_{i,\theta_i}^{cm} \le t_{i,\theta_i}^{total} - \tau | T_{i,\theta_i}^{cp} = \tau\right] \mathbb{P}\left[T_{i,\theta_i}^{cp} = \tau\left(I(\kappa)\frac{CD}{f_{CPU}}\right)\right] \tag{6.27}$$

Proof. See Appendix 6.C. □

6.4 Problem Formulation

The server aims at selecting suitable beamwidths for all UAVs that achieve the following conflicting objectives: (i) maximizing the network coverage to incorporate the data of more sensors in FL and (ii) minimizing the time duration of a global iteration.

6.4.1 Objective 1: Maximizing Coverage

One objective is to maximize the number of sensors uniquely covered by each UAV. That means maximizing the number of active sensors covered by the UAVs while minimizing the overlap between the UAVs' coverage regions. The problem is challenging as the sensors' distribution is unknown, and the number of active sensors is a random variable due to the random nature of energy harvesting. Formally,

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \ \sum_{i=1}^{U} N_i(\theta_i) - N_{\text{OL}}(\boldsymbol{\theta}), \tag{6.28}$$

where $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_U\}$ is the set of beamwidths assigned to the UAVs by the server. $N_i(\theta_i)$ is the number of active sensors covered by UAV i when it has beamwidth θ_i including the overlapping active sensors that are simultaneously covered by other UAVs. N_{OL} is the number of times active sensors are covered simultaneously by more than one UAV.

6.4.2 Objective 2: Minimizing the Maximum Delay Time

The second objective is to minimize the duration time of one global iteration, as described in Section 6.3. In synchronized FL, the server updates the global model only after receiving the local models of all UAVs. Therefore, the aim is to minimize the maximum update time in a global iteration. That is,

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \left(\max_{i \in \mathcal{U}} T^{\text{total}}_{i,\theta_i} \right). \tag{6.29}$$

6.5 The Bi-Objective MAB Model

In realistic scenarios, there is often no a priori information about the sensors' distribution, activity, or the channel quality. Therefore, we propose solving the problem using a bi-objective MAB model.

In the proposed model, the decision-maker (bandit agent) is the server, and the action set is the set of UAVs' beamwidth values. Thus, at every round, the server selects one configuration (arm) for the UAVs from a total of K possible configurations, $\boldsymbol{\theta}_k = [\theta_{1,k}, \theta_{2,k}, \ldots, \theta_{U,k}]$ where $\theta_{i,k}$ is the beamwidth of UAV i when the configuration k is selected. Consequently,

the server receives a reward vector $\boldsymbol{X}(\boldsymbol{\theta}_k) = [X^{\text{cov}}(\boldsymbol{\theta}_k), X^{\text{delay}}(\boldsymbol{\theta}_k)]$ with two elements, one for each objective. The first element of the reward vector concerns coverage so that

$$X^{\text{cov}}(\boldsymbol{\theta}_k) = \left(\sum_{i=1}^{U} N_i(\theta_{i,k}) - N_{\text{OL}}(\boldsymbol{\theta}_k) \right)_{\text{normalized}}. \tag{6.30}$$

The second element is related to the total delay, that is,

$$X^{\text{delay}}(\boldsymbol{\theta}_k) = \left(\frac{1}{\max\limits_{i \in \mathcal{U}} T^{\text{total}}_{i,\theta_{i,k}}} \right)_{\text{normalized}}. \tag{6.31}$$

Let $\boldsymbol{\mu}_k = [\mu_k^1, \mu_k^2]$ be the expected reward vector of arm k. The BO-MAB algorithm must find the optimal arms that maximize the expected reward vector $\boldsymbol{\mu}_k$.
The possible values for $\theta_i \in [\theta_{\min}, \theta_{\max}]$ are bounded from above, $\theta_{\max}$, by the minimum received power at the UAV, as given by (6.11). Larger upper bounds result in larger coverage areas and a lower received power. At the same time, the difference between the upper- and lower limit determines the level of heterogeneity in the network, which affects the FL performance. In practical systems, θ_i takes values on a discrete set of values. Let W be the number of beamwidths that the server can select for one UAV; then the total number of arms is

$$K = W^U. \tag{6.32}$$

For example, in a network with three UAVs, each with two possible angles, the server has $2^3 = 8$ possible configuration choices.

6.6 Solution for the Bi-Objective MAB Problem

A BO-MAB problem has multiple optimal solutions according to the specified reward vector. The set of reward vectors that are non-dominated by any other reward vectors is the *Pareto optimal reward set* $\mathcal{O}^*$. The arms whose reward vectors belong to $\mathcal{O}^*$ form the *Pareto optimal arm set* $\mathcal{A}^*$ (Drugan and Nowe, 2013). We use two methods to find optimal solutions for the beamwidth selection problem in Section 6.5: the Pareto and the scalarized upper-confidence-bound 1 (UCB1).

6.6.1 Pareto UCB1

This algorithm directly maximizes the reward vectors in the two-objective reward space. First, the agent pulls each arm once for initialization. Then, at each iteration, it finds the Pareto optimal arm set $\mathcal{A}'$ that corresponds to the optimal rewards. For all non-optimal

arms $\alpha \notin \mathcal{A}^{'}$, there exists a Pareto optimal arm $k \in \mathcal{A}^{'}$ that dominates arm α, i.e.,

$$\bar{\boldsymbol{X}}_{\alpha} + \sqrt{\frac{2\ln(n\sqrt[4]{Q|\mathcal{A}^{*}|})}{n_{\alpha}}} \not\succ \bar{\boldsymbol{X}}_{k} + \sqrt{\frac{2\ln(n\sqrt[4]{Q|\mathcal{A}^{*}|})}{n_{k}}} \tag{6.33}$$

where $\bar{\boldsymbol{X}}_{\alpha}$ is the empirical mean reward of arm α, the term $\sqrt{\frac{2\ln(n\sqrt[4]{Q|\mathcal{A}^{*}|})}{n_{\alpha}}}$ is the confidence interval for arm α, n_{α} is the number of pulls of arm α, and Q is the number of objectives, $Q = 2$ here. Since the number of Pareto optimal arms is not known to the player a priori, the term $\sqrt[4]{Q|\mathcal{A}^{*}|}$ in (6.33) can be replaced with $\sqrt[4]{QK}$ (Drugan and Nowe, 2013). The notation $\boldsymbol{u} \not\succ \boldsymbol{v}$ indicates that the vector $\boldsymbol{v}$ is non-dominated by $\boldsymbol{u}$, meaning that there exists at least one dimension q for which $v^{q} > u^{q}$. After finding the set of optimal arms $\mathcal{A}^{'}$, the algorithm pulls an arm $k \in \mathcal{A}^{'}$ uniformly at random and observes its reward. It then updates its average reward vector $\bar{\boldsymbol{X}}_{\alpha}$ and the counters. The process continues until meeting the stopping condition by finding the set of optimal arms.

The performance of the algorithm is measured by the Pareto regret; a measure of the distance between a sub-optimal reward vector $\boldsymbol{\mu}_{\alpha}$ and the Pareto optimal rewards. The Pareto regret of a sub-optimal arm α is the minimum positive value ε_{α}, denoted ε^{*}_{α}, that, when added to both objectives, results in a virtual optimal reward vector $\boldsymbol{v}^{*}_{\alpha}$ that belongs to the Pareto optimal reward set, i.e, $\boldsymbol{v}^{*}_{\alpha} = \boldsymbol{\mu}_{\alpha} + \boldsymbol{\varepsilon}^{*}_{\alpha} \in \mathcal{O}^{*}$ (Drugan and Nowe, 2013). The computational complexity for each round of the Pareto UCB1 algorithm is $\mathcal{O}(K\log(K))$ (Deb, 2011), this is to find the Pareto set in a bi-objective problem. The overall complexity of the algorithm is $\mathcal{O}(nK\log(K))$.

Algorithm 6.2 Pareto UCB1 (Drugan and Nowe, 2013)

1: Play each arm once, $n \leftarrow K$, $n_k \leftarrow 1, \forall k$
2: **repeat**
3: Find the Pareto set $\mathcal{A}^{'}$ such that $\forall k \in \mathcal{A}^{'}$ and $\forall \alpha$, (6.33) holds.
4: Pull arm k chosen uniform randomly from the set $\mathcal{A}^{'}$.
5: $n \leftarrow n+1$, $n_k \leftarrow n_k + 1$.
6: Update $\bar{\boldsymbol{X}}_k$.
7: **until** stopping condition is met.

6.6.2 Scalarized Bi-Objective UCB1

The scalarized bi-objective UCB1 method transfers a bi-objective reward into a single-objective using a scalarization function. To find the Pareto front, a set of S scalarization functions is used to obtain the different elements in the Pareto front set. A widely-

considered choice is the linear scalarization (Lin-S) function,

$$f^j(\boldsymbol{\mu}_k) = \omega_j^1 \mu_k^1 + \omega_j^2 \mu_k^2, \quad \forall k \in \mathcal{A}, j \in S, \tag{6.34}$$

where ω_j^1, ω_j^2 are the weights assigned to the first and second objectives under scalarization j, such that $\omega_j^1 + \omega_j^2 = 1$. The linear function is simple to implement but is incapable of finding all the optimal arms in a non-convex Pareto set (Drugan and Nowe, 2013, Miettinen, 1999). Alternatively, the Chebychev scalarization (Cheb-S) function can, under some conditions, find all the optimal arms in a non-convex Pareto set (Miettinen, 1999, Drugan and Nowe, 2013). The Chebychev scalarization function is given by

$$f^j(\boldsymbol{\mu}_k) = \min_{q \in \{1,2\}} \quad \omega_j^q(\mu_k^q - z^q) \quad \forall k \in \mathcal{A}, j \in S, \tag{6.35}$$

the reference point $z^q = \min_{1 \le k \le K} \quad \mu_k^q - \varepsilon^q$, where ε^q is a small positive value, $\varepsilon^q > 0$.

For a given scalarization function f^j, the optimum reward $\boldsymbol{\mu}^*$ maximizes $f^j(\boldsymbol{\mu}_k)$, i.e.,

$$f^j(\boldsymbol{\mu}^*) := \max_{1 \le k \le K} f^j(\boldsymbol{\mu}_k). \tag{6.36}$$

Algorithm 6.3 summarizes the scalarized bi-objective UCB1. For each of the scalarization functions f^j, the agent plays each arm once for initialization. It then updates the counters n^j for the number of times f^j is played, and n_k^j for the number of times arm k is played under scalarization j. Afterward, it selects a function f^j uniformly at random, and plays the arm k^* that maximizes the UCB term $f^j(\bar{\boldsymbol{X}}_k) + \sqrt{2\ln(n^j)/n_k^j}$. Next, it updates the average reward of arm k, $\bar{\boldsymbol{X}}_k$ and the counters.
The regret for scalarization function f^j and sub-optimal arm α is the *scalarized regret*,

$$\Delta_\alpha^j := \max_{1 \le k \le K} f^j(\boldsymbol{\mu}_k) - f^j(\boldsymbol{\mu}_\alpha). \tag{6.37}$$

However, this metric does not show the reduction in the bi-objective regret because it combines a set of independent regrets. Thus, we use the unfairness regret to incorporate the mean rewards of all optimal arms (Drugan and Nowe, 2013)

$$\mathfrak{R}_f(n) = \frac{1}{|\mathcal{A}^*|} \sum_{k \in \mathcal{A}^*} (T_k^*(n) - \mathbb{E}[T^*(n)])^2, \tag{6.38}$$

where $T_k^*(n)$ is the number of pulls of arm k and $\mathbb{E}[T^*(n)]$ is the expected number of times the optimal arms are pulled. The computational complexity for each scalarization function and each round is $\mathcal{O}(K)$. Let n be the number of times a scalarization function $f^j \in \{f^1, \ldots, f^S\}$ is sampled, then the total complexity is $\mathcal{O}(KSn)$.

Algorithm 6.3 Scalarized Bi-Objective UCB1 (Drugan and Nowe, 2013)

Input: $\mathcal{S} = \{f^1, f^2, \ldots, f^S\}$
for $j = 1 : S$ **do**
  Initialize the arms.
  Play each arm once and observe the reward $\boldsymbol{X}_k^j$.
  Update $n^j \leftarrow n^j + 1, n_k^j \leftarrow n_k^j + 1, \bar{\boldsymbol{X}}_k^j$.
end for
repeat
  Choose a function f^j from $\mathcal{S}$ uniformly at random.
  Pull arm k^* that maximizes $f^j(\bar{\boldsymbol{X}}_k) + \sqrt{2\ln(n^j)/n_k^j}$.
  Update $\bar{\boldsymbol{X}}_{k^*}$, and set $n_{k^*}^j \leftarrow n_{k^*}^j + 1, n^j \leftarrow n^j + 1$.
until stopping condition is met.

Remark 6.1 *The bandit algorithms used Sections 6.6 and **6.7** are extendable to dynamic environments with non-stationary statistical distributions (Balef and Maghsudi, 2023).*

6.7 Sequential Elimination For Scalarization-Based Best Arm Identification

Section 6.6 discussed two methods for solving the bi-objective optimization problem under uncertainty. However, those methods can be inefficient for a large number of UAVs or beamwidths due to the exponential growth in the number of arms, as given in (6.32). Therefore, this section proposes a novel solution using a generalized BAI algorithm that can control the number of arms eliminated at each round, allowing it to scale well for several arms. To ensure the energy efficiency of the proposed solution, it aims at finding the arm that maximizes the ratio of its expected reward to its expected energy cost. Furthermore, as this algorithm separates the exploration phase from the exploitation phase (pure exploration), the amount of UAV energy consumed during exploration is controlled based on the UAVs' energy budget.

The proposed algorithm extends the general sequential elimination algorithm for best arm identification in Shahrampour et al. (2017) to (i) include the scalarization-based bi-objective MAB algorithm, and (ii) find the arm that maximizes the ratio between the expected reward to the expected cost. The algorithm studies a fixed budget setting, where the agent aims to find the optimal arm with a high probability in a pre-determined number of rounds (budget). It is significant as it generalizes the successive rejects (Succ-Rej) algorithm for identifying the best arm (Audibert and Bubeck, 2010) by eliminating one arm in each round of the algorithm. It is also a generalization of the sequential halving (Seq-Halv) algorithm that removes half of the remaining arms in each round (Shahrampour

et al., 2017). Besides, *Shahrampour et. al.* develop a new algorithm, namely, *non-linear sequential elimination* (N-Seq-EL), which extends the Succ-Rej algorithm. Both algorithms eliminate one arm in every round, but the latter divides the remaining budget by a non-linear function of the remaining arms, a trick that can improve the best arm identification probability.
With each arm selection, the server receives $\boldsymbol{X}_k = [X^{\text{cov}}(\boldsymbol{\theta}_k), X^{\text{delay}}(\boldsymbol{\theta}_k)]$ a two-dimensional reward vector as defined in (6.30) and (6.31), with expected value $\boldsymbol{\mu}_k^X$. Also, this arm selection results in random energy cost c_k, with mean value μ_k^c, affecting both objectives equally. This cost represents the UAV's consumed energy in updating their local models and transmitting their local updates to the server. It is given by

$$c_k = \sum_{i=1}^{U} E_{i,\theta_i}^{\text{cp}} + E_i^{\text{cm}}, \tag{6.39}$$

where $E_{i,\theta_i}^{\text{cp}}$ is given in (6.15), $E_i^{\text{cm}} = P_{\text{t,G}} T_{i,\text{L}}$, and $P_{\text{t,G}}$ is the transmit power of the UAV. The server aims to find a beamwidth configuration (arm) $k \in \mathcal{K}$ that maximizes the ratio of the expected reward to the expected energy cost,

$$k^* = \underset{k \in \{1,\ldots,K\}}{\arg\max} \frac{f^j(\boldsymbol{\mu}_k^X)}{\mu_k^c}. \tag{6.40}$$

Algorithm 6.4 summarizes the scalarized cost-aware general sequential elimination (S-C-G-Seq-El) method. It starts by estimating the budget $n^{'}$, i.e., the number of rounds for all scalarization functions, based on the UAV's energy budget. Let E_{total} be the energy available at a UAV for learning and transmission, i.e., exploration. The fixed budget for running the algorithm is upper bounded by

$$n^{'} = \frac{E_{\text{total}}}{|\mathcal{S}| \left(\frac{\alpha}{2} N_{\theta_{\max}}^{\text{total}} DI(\kappa) \delta f_{\text{CPU}}^2 + P_{\text{t,G}} T_{\text{L,max}} \right)}, \tag{6.41}$$

where $N_{\theta_{\max}}^{\text{total}}$ is the number of sensors in the largest coverage area of a UAV, and $T_{\text{L,max}}$ is the maximum allowed upload time. The denominator in (6.41) represents the maximum energy consumed by a UAV. It occurs when all the sensors in its coverage region are active and the upload delay is maximized.
After estimating the budget, the S-C-G-Seq-El algorithm proceeds to find the optimal arms $\boldsymbol{I}_S^*$ from the Pareto front by identifying the optimal arm for each scalarization function in the set $\mathcal{S} = \{f^1, \ldots, f^j, \ldots, f^{|S|}\}$. Initially, the set of optimal arms $\boldsymbol{I}_S^*$ is empty. Besides, the set of active arms for each scalarization function f^j is the set of all arms $\mathcal{K}$, formally, $\boldsymbol{G}_1^j \leftarrow \mathcal{K}$. Then for each scalarization function, the algorithm runs for R rounds, where $R = K$ for successive rejects and $R = \lceil \log_2 K \rceil$ for sequential halving. In each round

ρ, it pulls each arm k in the set of active arms $\boldsymbol{G}_\rho^j$ for $n_\rho - n_{\rho-1}$ times,

$$n_\rho = \left\lceil \frac{n-K}{Cz_\rho} \right\rceil, \quad \rho = 1,\ldots,R, \tag{6.42}$$

where $n = \lfloor n^{'}/|S| \rfloor$ is the budget for each scalarization function, with $n^{'}$ being the total budget. Besides, $\{z_\rho\}_{\rho=1}^R$ is a decreasing positive sequence, and C is a constant, $C = \frac{1}{z_R} + \sum_{\rho=1}^R \frac{b_\rho}{z_\rho}$ (Shahrampour et al., 2017).

After pulling the arms, the algorithm discards the subset $\boldsymbol{B}_\rho^j$ of b_ρ arms with the lowest ratio of expected reward to the expected cost. It also updates the set of active arms, $\boldsymbol{G}_{\rho+1}^j = \boldsymbol{G}_\rho^j \backslash \boldsymbol{B}_\rho^j$. The process continues for R rounds to eliminate $K-1$ arms. The remaining arm k^* is the optimal one for the scalarization function f^j, so the algorithm includes it in the set of optimal arms $\boldsymbol{I}_S^*$. The cycle repeats for all the scalarization functions to identify all optimal arms. The computational complexity of the S-C-G-Seq-El algorithm is $\mathcal{O}(SRn)$.

Algorithm 6.4 The Scalarized Cost-Aware General Sequential Elimination Algorithm (S-C-G-Seq-El)

Input: Set of arms $\mathcal{K}$, set of scalarization functions $\mathcal{S}$, budget $n^{'}$, sequence $\{z_\rho, b_\rho\}_{\rho=1}^R$.
Initialization: $\boldsymbol{I}_S^* \leftarrow \phi, \boldsymbol{G}_1^j \leftarrow \mathcal{K}, \forall f^j \in S$.
Let $n_0 = 0$,
$C = \frac{1}{z_R} + \sum_{\rho=1}^R \frac{b_\rho}{z_\rho}$.
$n_\rho = \left\lceil \frac{n-K}{Cz_\rho} \right\rceil$ for $\rho = 1,\ldots,R$.
for all $f^j \in \mathcal{S}$ **do**
  for all rounds $\rho = 1,\ldots,R$ **do**
    Select each arm $k \in \boldsymbol{G}_\rho^j$ for $n_\rho - n_{\rho-1}$ times.
    Find the set of b_ρ arms with the smallest ratio of the expected reward to the expected cost, i.e., $\arg\min_{k \in \boldsymbol{G}_\rho^j} f^j(\bar{\boldsymbol{X}}_{k,n_\rho})/\bar{c}_{k,n_\rho}$, denoted by $\boldsymbol{B}_\rho^j$.
    Discard the worst b_ρ arms. Let $\boldsymbol{G}_{\rho+1}^j = \boldsymbol{G}_\rho^j \backslash \boldsymbol{B}_\rho^j$.
  end for
  if $k^* \notin \boldsymbol{I}_S^*$ where $k^* = \boldsymbol{G}_{R+1}^j$ **then**
    $\boldsymbol{I}_S^* \leftarrow \boldsymbol{I}_S^* \cup \{k^*\}$.
  end if
end for

Theorem 6.1 *The probability of misidentifying a sub-optimal arm as the optimal one for*

a scalarization function f^j is

$$\mathbb{P}\left[\boldsymbol{G}_{R+1}^{j} \neq \{k^*\}\right] \leq \max_{\rho\in\{1,\ldots,R\}}\{b_\rho\} \max_{\rho\in\{1,\ldots,R\}}\left\{2\exp\left(-2n_\rho\Xi_\rho^2\right)\right\}. \tag{6.43}$$

Proof. See Appendix 6.D □

6.8 Numerical Results

We consider a 1 km × 1 km square-shaped sensor network. The sensors' distribution is a Poisson point process with density $\lambda = 150$ sensor/km^2. The ground server is at the center. The locations of the four available UAVs are determined by the circle packing theorem. The UAVs' altitude is $H = 230$ m, which is optimal for the circle packing radius $R = 250$ m (Al-Hourani et al., 2014). In this example, we use the MNIST dataset for image classification for the FL training in the 4-UAV network. Each UAV collects the MNIST data samples from the active sensors in its coverage region and engages in a FL image classification task. We adapt the FL convolutional neural network parameters from reference (McMahan et al., 2017). Table 6.2 lists the parameters for the urban environment and transmission.

Parameter	Value	Description
λ	150 $\frac{\text{sensor}}{\text{km}^2}$	Sensor density
P_t	-3 dBW	Transmit power for all sensors and UAVs
$P_{\min}$	-100 dB	Minimum receive power at a UAV
H	230 m	UAV atitude
$B_{\text{G}}, B_{\text{F}}$	1 MHz	Channel bandwidth
a	9.61	Urban environment S-curve parameter
b	0.16	Urban environment S-curve parameter
η_{LoS}	1 dBW	Mean additional LoS losses
η_{NLoS}	20 dBW	Mean additional
f_c	2.5 GHz	Transmission frequency
α	2×10^{-28}	Capacitance coefficient
E_{total}	20 J	UAV's learning and transmission energy

Table 6.2: Simulation parameters.

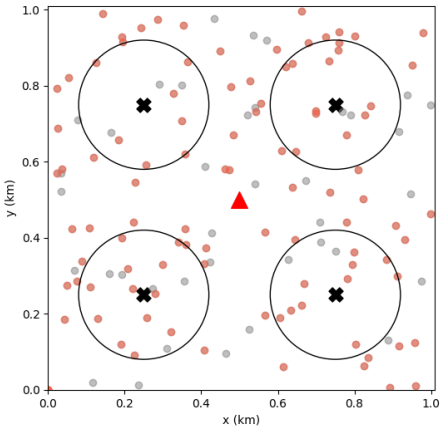
The wireless sensors have 5 cm^2 photovoltaic panels for harvesting solar energy. Given that the energy arrival rate on a sunny day is 15 mW/cm^2 (Grossi, 2021), each sensor harvests energy at a rate of 75mW. A sensor will transmit using the harvest-then-use energy harvesting scheme if it harvests at least $E_T = 100$ mJ and is covered by a UAV.

Each UAV can take two beamwidths, namely, $\theta = 41^\circ$ and 47.39°, which correspond to coverage radii 200 m and 250 m, respectively. Thus, the bi-objective MAB problem has $2^4 = 16$ arms, arm 1 has the configuration $\boldsymbol{\theta}_1 = \{41^\circ, 41^\circ, 41^\circ, 41^\circ\}$, arm 2 represents $\boldsymbol{\theta}_2 = \{41^\circ, 41^\circ, 41^\circ, 47.39^\circ\}$ and so on. Fig. 6.2 illustrates the simulated network for four different arms, i.e., beamwidth configurations.
Let $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{16}$ be the normalized average reward vectors of the 16 arms. A reward vector $\boldsymbol{\mu}_k$ dominates vector $\boldsymbol{\mu}_\alpha$ if $\boldsymbol{\mu}_k$ is not worse than $\boldsymbol{\mu}_\alpha$ in all objectives and $\boldsymbol{\mu}_k$ is strictly better than $\boldsymbol{\mu}_\alpha$ in at least one objective. The reward vectors that are not dominated by any other reward vector are optimal and they form the Pareto front. For benchmarking, we find the optimal reward set $\mathcal{O}^*$ and the corresponding arm set $\mathcal{A}^*$ by finding the empirical normalized average rewards of all arms over 5×10^6 iterations. Then, the set of non-dominated arms is found by exhaustive search, i.e., by comparing the arms' reward vectors to find the dominance relations. The same Pareto front was obtained using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). Fig. 6.3a shows the normalized average reward for all the arms, the set of non-dominated average rewards $\mathcal{O}^*$ corresponds to the optimal arm set $\mathcal{A}^* = \{1, 3, 7, 8, 15, 16\}$. To interpret this result in terms of the network parameters, Table 6.3 transforms the normalized average reward of each arm to the corresponding average number of covered sensors $\bar{N}_{\boldsymbol{\theta}}$ and the average maximum delay denoted by $\bar{T}_{\max}$. Clearly, arms 1 and 16 are optimal because they minimize the maximum delay and maximize the coverage, respectively. Arms 3, 7 ,8, and 15 achieve a tradeoff between the two objectives. However, arm 9, for example, is not optimal since arm 7 achieves a higher coverage and less delay.

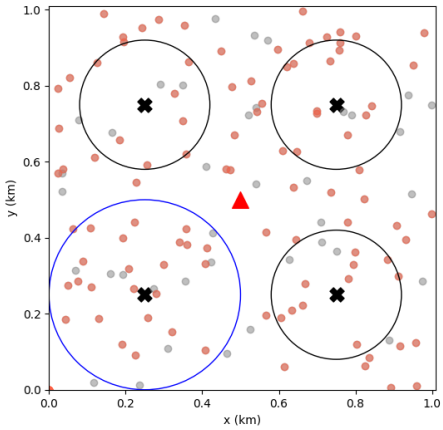
Arm	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\bar{N}_{\boldsymbol{\theta}}$	55.5	62.6	63.1	71.1	62.1	69.1	68.8	76.1	61.3	68.4	68.8	76.8	67.5	74.4	75.4	83.1
$\bar{T}_{\max}$ (s)	1.13	1.31	1.16	1.31	1.21	1.32	1.22	1.32	1.28	1.35	1.30	1.35	1.30	1.35	1.30	1.35

Table 6.3: Arm's coverage and maximum delay.

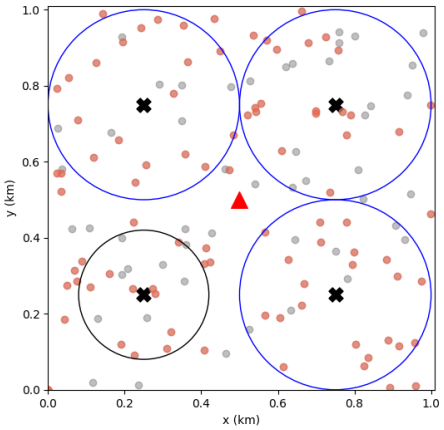
Fig. 6.3b shows the fraction of time each arm is played in 5×10^6 rounds. Pareto UCB1 pulls the optimal arms fairly. For the scalarized bi-objective UCB1 algorithm, we use 11 uniformly spaced weights, $\boldsymbol{\omega}^j \in \{[1,0], [0.9,0.1], \ldots, [0,1]\}$. The LinS method cannot find all the optimal arms in the non-convex Pareto front because there is no set of weights that describes all optimal reward vectors. Consequently, the algorithm exploits arms 1 and 16 significantly more frequently than the other optimal arms. The ChebS performs better than LinS in exploiting the optimal arms as it can find the optimal arms in a non-convex Pareto front. This fact also appears in terms of the unfairness regret in Fig. 6.3c. Fig. 6.3d shows the average regret. The Pareto UCB1 algorithm has a low average regret as it finds all optimal arms and plays them fairly. In contrast, the LinS algorithm incurs a large average regret as it fails to find all the optimal arms. ChebS has a lower average regret than LinS as it performs better in finding the optimal arms.



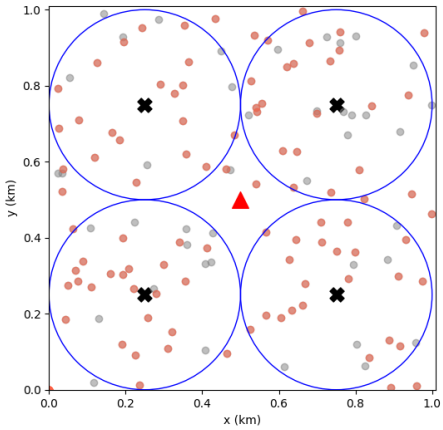
(a) Arm 1 represents configuration $\boldsymbol{\theta}_1 = \{41°, 41°, 41°, 41°\}$



(b) Arm 2 represents $\boldsymbol{\theta}_2 = \{41°, 41°, 41°, 47.39°\}$

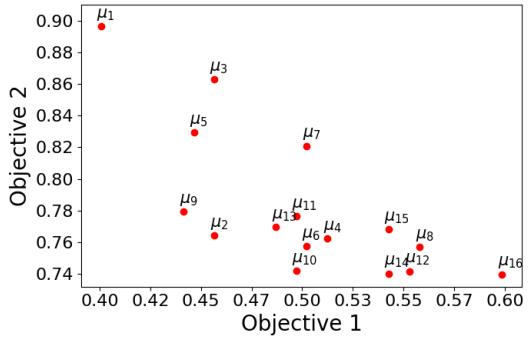


(c) Arm 15 represents $\boldsymbol{\theta}_{15} = \{47.39°, 47.39°, 47.39°, 41°\}$

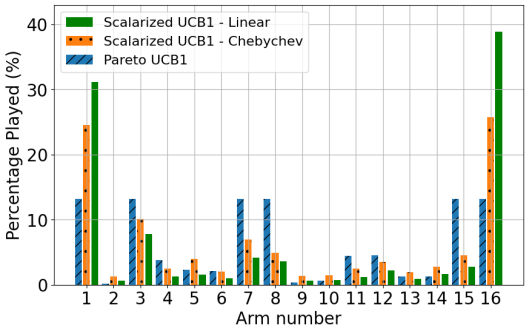


(d) Arm 16 represents $\boldsymbol{\theta}_{16} = \{47.39°, 47.39°, 47.39°, 47.39°\}$

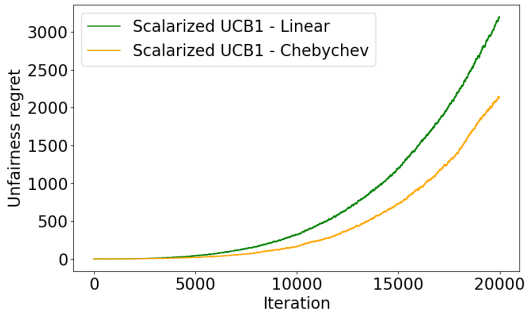
Figure 6.2: An illustration of the network, the red triangle represents the ground server, and the circles represent the coverage area of each of the UAVs. The red and gray dots represent the active and inactive sensors, respectively.



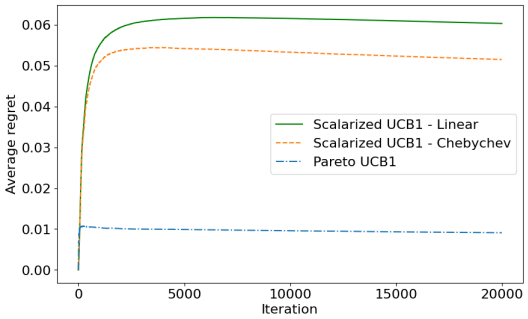
(a) The normalized average reward for all arms.



(b) The percentage of playing each arm.



(c) The unfairness regret of the Lin-S and Cheby-S UCB1.



(d) The average regret.

Figure 6.3: The performance of the 4-UAV network.

The multi-objective optimization problem returns a set of optimal arms. However, the server selects one of them as the UAVs' beamwidths based on its preference and the application. To show this, we consider a 4-UAV network that is similar to the one described above, with the exception that the sensors' locations follow an inhomogeneous Poisson distribution with density $\lambda(x,y) = 150(x^2+y^2)$ sensors/km^2. Among the 16 arms, six are optimal. Fig. 6.4 shows the optimal arms with their coverage and maximum delay. As expected, the circle packing placement (arm 16) maximizes coverage; nevertheless, it might not fit the FL process as it prolongs the delay. Generally, using any of these 6 arms is optimal as opposed to the sub-optimal arms.

The next example shows the performance of the S-C-G-Seq-El algorithm using the network of the first numerical example and linear scalarization. The S-C-G-Seq-El algorithm starts by estimating the budget based on the maximum energy consumed by the UAVs per global round when the UAVs take the largest beamwidths. Then, the optimal arms are obtained by finding the best arm for each scalarization function. The performance analysis here is more complicated than the single objective case (Audibert and Bubeck, 2010, Shahrampour et al., 2017) because there are multiple performance measures.
In Fig. 6.5a -Fig. 6.5c, we aim to find the arms that maximize the expected reward to the ex-

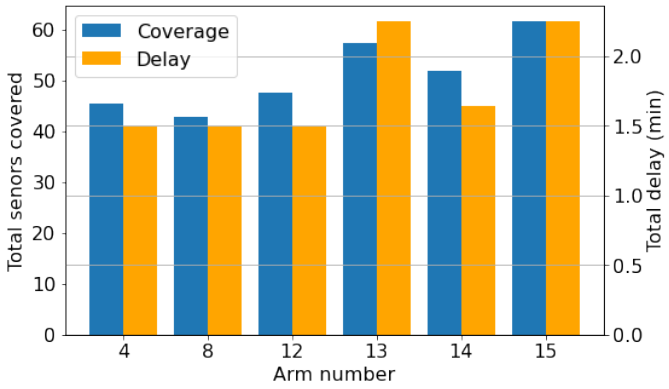
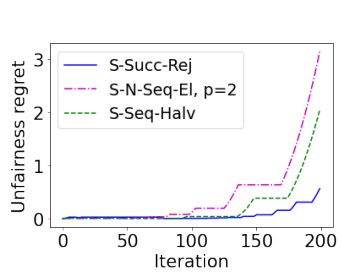
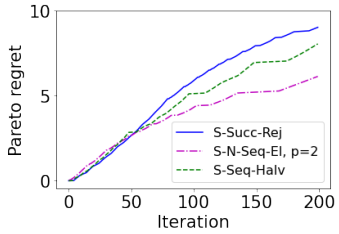


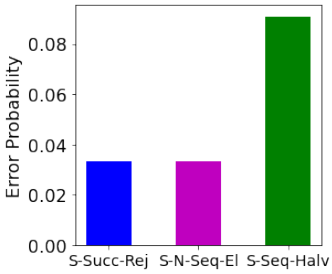
Figure 6.4: The sensor coverage and total delay at the optimal beamwidth configurations.



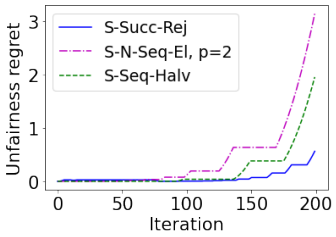
(a) The unfairness regret when the arms' energy cost is considered.



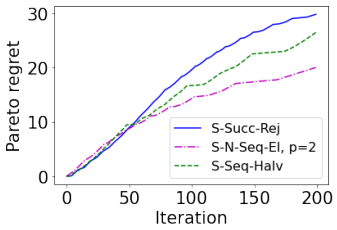
(b) The Pareto regret when the arms' energy cost is considered.



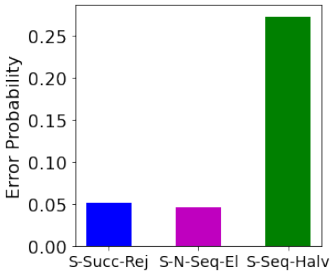
(c) The error probability when the arms' energy cost is considered.



(d) The unfairness regret for arms with unit cost.



(e) The Pareto regret for arms with unit cost.



(f) The error probability for arms with unit cost.

Figure 6.5: The unfairness regret, average regret, and error probability for different instances the S-C-G-Seq-El algorithm for with and without considering the energy cost.

pected cost ratio for three instances of the S-C-G-Seq-El: S-Succ-Rej, S-N-Seq-EL with $p = 2$, and S-Seq-Halv using linear scalarization. The prefix (S-) stands for scalarization to indicate that this is a bi-objective problem. First, we obtained the optimal arms under linear scalarization and the 11 uniformly spaced weights, $\boldsymbol{\omega}^j \in \{[1,0],[0.9,0.1],\ldots,[0,1]\}$, the optimal arms were arms 1, 3 and 16. Fig. 6.5a shows the unfairness regret. Although

both algorithms eliminate one arm every round, the S-Succ-Rej has a lower unfairness regret than S-N-Seq-EL. This is because, in the first rounds when most arms are still available, S-Succ-Rej plays the arms longer than S-N-Seq-El, leaving less budget for the following rounds which results in higher fairness. Fig. 6.5b shows the instantaneous Pareto regret given by $\Delta_k^j = \frac{f^j(\bar{\boldsymbol{X}}_{k^*})}{\bar{c}_{k^*}} - \frac{f^j(\bar{\boldsymbol{X}}_k)}{\bar{c}_k}$, where k^* is the optimal arm under f^j, averaged on all scalarization functions. The S-N-Seq-El with $p = 2$ has the lowest Pareto regret, followed by Seq-Halv which implies that eliminating half of the arms in each round does not necessarily result in a higher Pareto regret. Fig. 6.5c shows the probability of discarding the optimal arm for a given scalarization function, averaged over all scalarization functions. The three algorithms are effective in finding the optimal arms as they have a low error probability.
Our proposed algorithm considers the arms' energy cost to ensure that the recommended UAV configuration is energy efficient. But the algorithm can be used to find the arms that maximize the expected reward without considering the cost, i.e., when all arms have a unitary cost. Fig. 6.5d -Fig. 6.5f show the results for the latter case. The discussion here is similar to the previous case, except for higher Pareto regret and error probability. This is because when the arms' cost was considered, arm 1 had a much higher ratio of expected reward to expected energy cost compared to the other arms, and the algorithm was capable of finding it with a low error probability despite the limited budget. However, for the case of unitary cost, the sub-optimality gap was smaller, leading to more errors in finding the optimal arm for a given scalarization function f^j. Still, the low Pareto regret shows that the reward of the recommended arm is close to the optimal one.

6.9 Conclusion

In this chapter, the performance of FL in UAV-enhanced wireless networks was optimized under uncertainty about channels' quality, sensors' distribution, and transmission probability. Each UAV collects data from the sensors in its coverage regions and operates as an FL client. As the network parameters significantly affect the FL performance, we jointly optimize two conflicting objectives: maximizing the sensor coverage and minimizing the FL convergence time. We proposed a solution based on BO-MAB with two variations, Pareto UCB1, and linear multi-objective UCB1. Besides, we developed an efficient solution, S-C-G-Seq-El, based on the best arm identification concept. We define the best arm as one that maximizes the expected reward to the expected energy-cost ratio for an energy-efficient solution. Numerical results show that Pareto UCB1 performs better than the linearly-scalarized UCB1 algorithm in finding the optimal arms and playing them almost equally frequently. Then, we investigated the performance of three instances of the S-C-G-Seq-El algorithm. Although S-Seq-Halv removes half of the arms in every round, its performance concerning unfairness and Pareto regret is comparable to S-Succ-Rej and S-N-Seq-El, which eliminate one arm per round. Our proposed solution has a

wide range of applications in communication networks, as most optimization problems in such a network are intrinsically multi-objective. An example is client selection in FL to jointly optimize the convergence time, accuracy, and energy consumption.

Appendix

6.A Proof of Proposition 6.1

Proof. Let $\gamma_{i,\mathrm{G}} = \frac{P_{\mathrm{t,G}} G_{i,\mathrm{G}}}{L_{i,\mathrm{G}} \sigma_{\mathrm{G}}^2}$. The transmission rate of the UAV or the server is

$$r_{i,\mathrm{G}} = B_{\mathrm{G}} \log_2 \left(1 + \gamma_{i,\mathrm{G}} |g_{i,\mathrm{G}}|^2\right). \tag{6.44}$$

The time required to transmit a model of size S_{M} is given by

$$T_{i,\mathrm{G}}^{\mathrm{cm}} = \frac{S_{\mathrm{M}}}{r_{i,\mathrm{G}}}. \tag{6.45}$$

To obtain the distribution of the UAV-server communication time, we first find the distribution of the UAV-server data rate given in (6.44) and then use (6.45) to find the distribution of the communication time.
When $g_{i,\mathrm{G}}$ is Ricean distributed, then $\Psi_{i,\mathrm{G}} = |g_{i,\mathrm{G}}|^2$ follows the non-central χ^2 distribution with two degrees of freedom. Its PDF and CDF respectively are (Proakis and Salehi, 2008)

$$\begin{aligned} f_X(x) &= \frac{1}{2\xi^2} e^{-\frac{k^2+x}{2\xi^2}} I_0\left(\frac{k}{\xi^2}\sqrt{x}\right), \\ F_X(x) &= 1 - Q_1\left(\frac{k}{\xi}, \frac{\sqrt{x}}{\xi}\right), \quad x > 0, \end{aligned} \tag{6.46}$$

where ξ and k are the parameters of the χ^2 distribution. $I_0(x)$ is the modified Bessel function of the first kind and order zero, i.e., $I_0(x) = \frac{1}{2\pi}\int_0^{2\pi} e^{x\cos\phi} d\phi$, and $Q_1(.,.)$ is the Marcum Q function, given by $Q_1(a,b) = \int_b^{\infty} x e^{-\frac{a^2+x^2}{2}} I_0(ax) dx$. Thus, the distribution of Ψ_i yields

$$f_{\Psi_{i,\mathrm{G}}}(\psi_{i,\mathrm{G}}) = \frac{1}{2\xi_{\mathrm{G}}^2} e^{-\frac{k_{i,\mathrm{G}}^2 + \psi_{i,\mathrm{G}}}{2\xi_{\mathrm{G}}^2}} I_0\left(\frac{k_{i,\mathrm{G}}}{\xi_{\mathrm{G}}^2}\sqrt{\psi_{i,\mathrm{G}}}\right), \qquad \psi_{i,\mathrm{G}} > 0 \tag{6.47}$$

Now, the distribution of the data rate $r_{i,\mathrm{G}}$ in (6.44) can be obtained using the transformation of random variables. Besides, $\mathbb{P}[r_{i,\mathrm{G}} < 0] = 0$ because $\log_2(1+x) > 0$ for $x > 0$,

$$\begin{aligned}\mathbb{P}[r_{i,\mathrm{G}} \leq \rho_{i,\mathrm{G}}] &= \mathbb{P}\left[B_{\mathrm{G}} \ln\left(1+\gamma_i \psi_i\right) \leq \rho_{i,\mathrm{G}}\right] \\ &= \mathbb{P}\left[\psi_{i,\mathrm{G}} \leq \frac{e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}{\gamma_{i,\mathrm{G}}}\right]. \end{aligned} \tag{6.48}$$

Now, substituting (6.48) in $F_X(x)$ from (6.46) results in the CDF of the transmission rate

$$F_{r_{i,\mathrm{G}}}(\rho_{i,\mathrm{G}}) = F_{\Psi_{i,\mathrm{G}}}\left(\frac{e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}{\gamma_{i,\mathrm{G}}}\right) = 1 - Q_1\left(\frac{k_{i,\mathrm{G}}}{\xi_{\mathrm{G}}}, \frac{\sqrt{e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}}{\xi_{\mathrm{G}}\sqrt{\gamma_{i,\mathrm{G}}}}\right). \tag{6.49}$$

The CDF can be rewritten as:

$$\begin{aligned} F_{r_{i,\mathrm{G}}}(\rho_{i,\mathrm{G}}) &= 1 - Q_1\left(\frac{k_{i,\mathrm{G}}}{\xi_{\mathrm{G}}}, \frac{\sqrt{e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}}{\xi_{\mathrm{G}}\sqrt{\gamma_{i,\mathrm{G}}}}\right) \\ &= 1 - \int_{\frac{\sqrt{e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}}{\xi_{\mathrm{G}}\sqrt{\gamma_{i,\mathrm{G}}}}}^{\infty} x e^{-\frac{\frac{k_{i,\mathrm{G}}^2}{\xi_{\mathrm{G}}^2} + x^2}{2}} I_0\left(\frac{k_{i,\mathrm{G}} x}{\xi_{\mathrm{G}}}\right) dx. \end{aligned} \tag{6.50}$$

To find the PDF of $r_{i,\mathrm{G}}$, we must find the derivative of the CDF given in (6.50). By applying the Leibniz rule (Papoulis and Pillai, 2002) to (6.50), we arrive at

$$f_{r_{i,\mathrm{G}}}(\rho_{i,\mathrm{G}}) = \frac{1}{2B_{\mathrm{G}}\gamma_{i,\mathrm{G}}\xi_{\mathrm{G}}^2} e^{\frac{k_{i,\mathrm{G}}^2\gamma_{i,\mathrm{G}} + e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}{2\gamma_{i,\mathrm{G}}\xi_{\mathrm{G}}^2}} I_0\left(\frac{k_{i,\mathrm{G}}\sqrt{e^{\frac{\rho_{i,\mathrm{G}}}{B_{\mathrm{G}}}} - 1}}{\xi_{\mathrm{G}}^2\sqrt{\gamma_{i,\mathrm{G}}}}\right). \tag{6.51}$$

The final step is to obtain the distribution of the UAV-server communication time using (6.45) by transforming the random variable. Let $y = g(x) = \frac{a}{x}$ with $x, a \geq 0$. The PDF of y yields $f_Y(y) = \frac{a}{y^2} f_X\left(\frac{a}{y}\right)$ (Papoulis and Pillai, 2002). Thus, the PDF of the communication time is given by

$$f_{T_{i,\mathrm{G}}}(t_{i,\mathrm{G}}) = \frac{S_{\mathrm{M}}}{2B_{\mathrm{G}}\gamma_{i,\mathrm{G}}\xi_{\mathrm{G}}^2 t_{i,\mathrm{G}}^2} e^{\frac{k_1^2\gamma_{i,\mathrm{G}} + e^{\left(\frac{S_{\mathrm{M}}}{B_{\mathrm{G}} t_{i,\mathrm{G}}}\right)} - 1}{2\gamma_{i,\mathrm{G}}\xi_{\mathrm{G}}^2}} I_0\left(\frac{k_{i,\mathrm{G}}\sqrt{e^{\frac{S_{\mathrm{M}}}{B_{\mathrm{G}} t_{i,\mathrm{G}}}} - 1}}{\xi_{\mathrm{G}}^2\sqrt{\gamma_{i,\mathrm{G}}}}\right), \tag{6.52}$$

which completes the proof. □

6.B Proof of Proposition 6.2

Proof. The sensors are distributed according to a Poisson distribution. Thus, their pairwise distance follows an exponential distribution, i.e., $f_{Z_i}(z_i) = \lambda \Omega_{i,\theta_i} e^{\left(\lambda \Omega_{i,\theta_i}\right) z_i}$.
The maximum distance between the sensor and a UAV is thus the maximum order statistics of the exponential distribution. The maximum order statistics of n independent samples of a continuous random variable is (Arnold et al., 2008) $f_{X_{(n:n)}}(x) = n\left[F_X(x)\right]^{n-1} f_X(x)$. Thus, the distribution of the maximum sensor distance within the coverage area of UAV i, Ω_{i,θ_i}, results from substituting $f_{Z_i}(z_i)$ in $f_{X_{(n:n)}}(x)$. That is,

$$f_{\bar{Z}_{i(\bar{N}_{i,\theta_i}:\bar{N}_{i,\theta_i})}}(z_i) = \lambda \bar{N}_{i,\theta_i} \Omega_{i,\theta_i} \left[1 - e^{\left(\lambda \Omega_{i,\theta_i}\right) z_i}\right]^{\bar{N}_{i,\theta_i}-1} \Omega_{i,\theta_i} e^{\left(\lambda \Omega_{i,\theta_i}\right) z_i}, \tag{6.53}$$

where $\bar{N}_{i,\theta_i} = \lambda \Omega_{i,\theta_i}$ represents the average number of active sensors in the coverage disk of UAV i. The average maximum distance is then

$$\begin{aligned} \bar{Z}_{i,\theta_i} &= \int_0^\infty z_i f_{Z_{i\bar{N}_{i,\theta_i}:\bar{N}_{i,\theta_i}}}(z_i) dz_i \\ &= \int_0^\infty z_i \lambda \bar{N}_{i,\theta_i} \Omega_{i,\theta_i} e^{\left(\lambda \Omega_{i,\theta_i}\right) z_i} \left[1 - e^{\left(\lambda \Omega_{i,\theta_i}\right) z_i}\right]^{\bar{N}_{i,\theta_i}-1} dz_i. \end{aligned} \tag{6.54}$$

□

6.C Proof of Proposition 6.4

Proof. The computation time $T_{i,\theta}^{\text{cp}}$ is a discrete random variable while the communication time $T_{i,\theta}^{\text{cm}}$ is continuous. They are dependent as both are a function of θ_i. Thus, the cumulative distribution function (CDF) of their sum $T_{i,\theta}^{\text{total}} = T_{i,\theta}^{\text{cp}} + T_{i,\theta}^{\text{cm}}$ is derived from the definition of the CDF, i.e.,

$$\begin{aligned} F_{T_{i,\theta_i}^{\text{total}}}\left(t_{i,\theta_i}^{\text{total}}\right) &= \mathbb{P}\left[T_{i,\theta_i}^{\text{total}} \leq t_{i,\theta_i}^{\text{total}}\right] \\ &= \mathbb{P}\left[t_{i,\theta_i}^{\text{cp}} + t_{i,\theta_i}^{\text{cm}} \leq t_{i,\theta_i}^{\text{total}}\right] \\ &= \sum_{\tau \in \bar{N}_{i,\theta_i}} \mathbb{P}\left[t_{i,\theta_i}^{\text{cm}} \leq t_{i,\theta_i}^{\text{total}} - \tau | T_{i,\theta_i}^{\text{cp}} = \tau\right] \mathbb{P}\left[T_{i,\theta_i}^{\text{cp}} = \tau\left(I(\kappa)\frac{CD}{f_{\text{CPU}}}\right)\right]. \end{aligned} \tag{6.55}$$

□

6.D Proof of Theorem 6.1

Proof. Each round ρ starts with a set $\boldsymbol{G}_\rho$ of g_ρ available arms. At the end of the round, the algorithm removes the set $\boldsymbol{B}_\rho$ of the worst b_ρ arms. Accordingly, an error occurs when the algorithm recommends a sub-optimal arm by the end of the R rounds (Shahrampour et al., 2017), i.e.,

$$\begin{aligned}\mathbb{P}\left[\boldsymbol{G}_{R+1} \neq \{k^*\}\right] &= \mathbb{P}[k^* \in \cup_{\rho=1}^{R} \boldsymbol{B}_\rho] \\ &= \sum_{\rho=1}^{R} \sum_{G_\rho} \mathbb{P}[k^* \in \boldsymbol{B}_\rho | \boldsymbol{G}_\rho = G_\rho] \mathbb{P}[\boldsymbol{G}_\rho = G_\rho]. \end{aligned} \tag{6.56}$$

The last equality holds because the sets of eliminated arms in different rounds are disjoint. Following the notation in Audibert and Bubeck (2010), we use $(k) \in \{1, \ldots, K\}$ to denote k-th best arm with the ties broken arbitrarily, thus $\frac{\mu^X_{(1)}}{\mu^c_{(1)}} \geq \frac{\mu^X_{(2)}}{\mu^c_{(2)}} \geq \cdots \geq \frac{\mu^X_{(K)}}{\mu^c_{(K)}}$. We also define the *sub-optimality gap* as $\Delta_k = \frac{\mu^X_{k^*}}{\mu^c_{k^*}} - \frac{\mu^X_k}{\mu^c_k}$. Thus, $\Delta_{(1)} = \Delta_{(2)} \leq \Delta_{(3)} \leq \cdots \leq \Delta_{(K)}$.
To find the upper bound on the error probability, we consider the worst case. It occurs when the round starts with the set of g_ρ best arms so that $G_\rho = \{(1), (2), \ldots, (g_\rho)\}$, and the optimal arm is eliminated with $B_\rho = \{(g_\rho - b_\rho + 1), \ldots, (g_\rho)\}$. Formally,

$$\frac{\bar{X}_{k^*,n_\rho}}{\bar{c}_{k^*,n_\rho}} \leq \max_{k \in \{(g_\rho - b_\rho + 1), \ldots, (g_\rho - 1), (g_\rho)\}} \frac{\bar{X}_{(k),n_\rho}}{\bar{c}_{(k),n_\rho}}. \tag{6.57}$$

For any other G_ρ the worst arm cannot be better than $(g_\rho - b_\rho + 1)$ (Shahrampour et al., 2017). Therefore, $\mathbb{P}(\boldsymbol{G}_{R+1} \neq \{k^*\}) \leq \sum_{\rho=1}^{R} \mathbb{P}(k^* \in \boldsymbol{B}_\rho | \boldsymbol{G}_\rho = \{(1), (2), \ldots, (g_\rho)\})$. Using the union bound, the probability of the event in (6.57) yields

$$\mathbb{P}\left[\boldsymbol{G}_{R+1} \neq \{k^*\}\right] \leq \sum_{\rho=1}^{R} \sum_{k=g_\rho - b_\rho + 1}^{g_\rho} \mathbb{P}\left[\frac{\bar{X}_{k^*,n_\rho}}{\bar{c}_{k^*,n_\rho}} \leq \frac{\bar{X}_{(k),n_\rho}}{\bar{c}_{(k),n_\rho}}\right]. \tag{6.58}$$

The event $\frac{\bar{X}_{k^*,n_\rho}}{\bar{c}_{k^*,n_\rho}} \leq \frac{\bar{X}_{(k),n_\rho}}{\bar{c}_{(k),n_\rho}}$ implies that at least one of the following two holds:

$$\frac{\bar{X}_{k^*,n_\rho}}{\bar{c}_{k^*,n_\rho}} \leq \varepsilon; \quad \frac{\bar{X}_{(k),n_\rho}}{\bar{c}_{(k),n_\rho}} \geq \varepsilon. \tag{6.59}$$

Accordingly, the error probability can be expressed as (6.60). By re-arranging the terms we obtain (6.62).

$$\mathbb{P}[\boldsymbol{G}_{R+1} \neq \{k^*\}] \leq \sum_{\rho=1}^{R} \sum_{k=g_\rho-b_\rho+1}^{g_\rho} \mathbb{P}\left[\frac{\bar{X}_{k^*,n_\rho}}{\bar{c}_{k^*,n_\rho}} \leq \varepsilon\right] + \mathbb{P}\left[\frac{\bar{X}_{(k),n_\rho}}{\bar{c}_{(k),n_\rho}} \geq \varepsilon\right] \tag{6.60}$$

$$= \sum_{\rho=1}^{R} \sum_{k=g_\rho-b_\rho+1}^{g_\rho} \mathbb{P}\left[\bar{X}_{k^*,n_\rho} - \varepsilon\bar{c}_{k^*,n_\rho} \leq 0\right] + \mathbb{P}\left[\bar{X}_{(k),n_\rho} - \varepsilon\bar{c}_{(k),n_\rho} \geq 0\right] \tag{6.61}$$

$$= \sum_{\rho=1}^{R} \sum_{k=g_\rho-b_\rho+1}^{g_\rho} \mathbb{P}\left[-(\bar{X}_{k^*,n_\rho} - \varepsilon\bar{c}_{k^*,n_\rho}) + (\mu^X_{k^*} - \varepsilon\mu^c_{k^*}) \geq \mu^X_{k^*} - \varepsilon\mu^c_{k^*}\right]$$
$$+ \mathbb{P}\left[(\bar{X}_{(k),n_\rho} - \varepsilon\bar{c}_{(k),n_\rho}) - (\mu^X_{(k)} - \varepsilon\mu^c_{(k)}) \geq -\mu^X_{(k)} + \varepsilon\mu^c_{(k)}\right]. \tag{6.62}$$

Note that at every selection round, the reward and the energy cost are dependent as they both depend on $\boldsymbol{\theta}_k$ and $T_{i,\mathrm{L}}$. However, the rewards obtained at different times are independent of each other, and so are the energy costs. Hence, the Hoeffding inequality still applies in (6.62).

$$\mathbb{P}[\boldsymbol{G}_{R+1} \neq \{k^*\}] \leq \sum_{\rho=1}^{R} \sum_{k=g_\rho-b_\rho+1}^{g_\rho} \exp\left(-\frac{2n_\rho\left(\mu^X_{k^*} - \varepsilon\mu^c_{k^*}\right)^2}{(\varepsilon+1)^2}\right)$$
$$+ \exp\left(-\frac{2n_\rho\left(-\mu^X_{(k)} + \varepsilon\mu^c_{(k)}\right)^2}{(\varepsilon+1)^2}\right). \tag{6.63}$$

By substituting the value of $\varepsilon = \frac{\mu^X_{k^*}+\mu^X_{(k)}}{\mu^c_{k^*}+\mu^c_{(k)}}$ in (6.63) and rearranging the terms, (6.63) reduces to

$$\mathbb{P}[\boldsymbol{G}_{R+1} \neq \{k^*\}] \leq \sum_{\rho=1}^{R} \sum_{k=g_\rho-b_\rho+1}^{g_\rho} 2\exp\left(-2n_\rho\left(\frac{\mu^X_{k^*}\mu^c_{(k)} - \mu^c_{k^*}\mu^X_{(k)}}{\mu^X_{k^*} + \mu^X_{(k)} + \mu^c_{k^*} + \mu^c_{(k)}}\right)^2\right). \tag{6.64}$$

Let $\Xi_\rho = \min_{k\in\{(g_\rho - b_\rho + 1),\ldots,(g_\rho)\}} \left(\frac{\mu^X_{k^*}\mu^c_{(k)} - \mu^c_{k^*}\mu^X_{(k)}}{\mu^X_{k^*} + \mu^X_{(k)} + \mu^c_{k^*} + \mu^c_{(k)}} \right)$, the upper bound on the error probability yields

$$\begin{aligned}\mathbb{P}[\boldsymbol{G}_{R+1} \neq k^*] &\leq \sum_{\rho=1}^{R} 2b_\rho \exp\left(-2n_k\Xi_\rho^2\right) \\ &\leq R \max_{\rho\in\{1,\ldots,R\}}\{b_\rho\} \max_{\rho\in\{1,\ldots,R\}}\left\{2\exp\left(-2n_\rho\Xi_\rho^2\right)\right\}. \end{aligned} \tag{6.65}$$

□

Chapter 7

Joint Coverage and Resource Allocation for Federated Learning in UAV-Enabled Networks

Note: This publication was presented at the IEEE Wireless Communications and Networking Conference (WCNC) 2022 (Yahya and Maghsudi, 2022).

Abstract Thanks to its communication efficiency and low latency, federated learning (FL) has emerged as a promising learning paradigm in the unmanned aerial vehicle (UAV)-enabled networks; nevertheless, the great potential of FL in UAV networks is realizable only upon optimizing crucial factors such as coverage and transmission delay. In this chapter, we study the problem of joint coverage optimization and efficient radio resource allocation. The objective is to minimize the convergence time of FL in a UAV-enabled network, where UAVs perform learning over an inhomogeneous sensor network. To this end, we develop a method that minimizes the FL computation and communication time in each global iteration: First, the algorithm adjusts the UAVs' locations to control the average number of sensors associated with each UAV to maximize the coverage and to reduce the overall computation time. The UAVs' locations also affect their transmission delay. Thus, in the second step, the method uses a fair resource allocation scheme for channel allocation and power control to minimize the FL communication time while retaining the efficiency of resource expenditure.

7.1 Introduction

Recently, unmanned aerial vehicle (UAV)-enabled networking has emerged as a new paradigm for managing the complexity of processing the excessive amount of data. To this end, UAVs cooperate in learning by taking advantage of their implemented machine learning algorithms (Brik et al., 2020). While centralized machine learning algorithms might be impractical due to the UAVs' mobility and the scarcity of radio resources, distributed learning is envisioned as a potential alternative.

Federated learning (FL) is a privacy-preserving distributed learning algorithm. In each iteration, upon receiving the global model from the server, every UAV updates the model using the data collected by the sensors in its coverage area, then transmits the locally updated models to the server. The server aggregates the local parameters and, based on them, generates a new global model. This process continues until the global model converges with a predetermined accuracy (Dinh et al., 2021).

FL preserves privacy and reduces the communication cost; nevertheless, the iterative updates and transmissions raise some challenges concerning convergence time and resource expenditure. The FL convergence time depends on the UAVs' computation times and the transmission delays between the UAVs and the server. These parameters, in turn, depend on the network design. Therefore, deploying UAVs to merely maximize the network coverage does not suffice to optimize the FL procedure. Moreover, the transmission delay is affected by the UAVs' placement and the radio resource allocation.

There are several approaches to optimizing the FL convergence time in resource-limited wireless networks. Some aim at reducing the data rate by implementing communication efficient compression algorithms (Sattler et al., 2020). Others leverage a client selection method where only a subset of clients receive the network's scarce resources for data transmission (Zhang et al., 2021). The work in (Yang et al., 2020) exploits the signal superposition property of multi-access channels to allocate the limited bandwidth to maximize the number of participants in the model update to accelerate the FL convergence. Some recent researches consider the convergence of UAV-enabled networks. Zeng et al. (2020), for example, proposes a novel framework for implementing FL within a UAV swarm. They propose a joint power allocation and UAV scheduling approach to optimize the convergence performance.

In this chapter, we develop a novel method for joint UAV placement and resource allocation. We aim to maximize the network coverage while satisfying the constraints on the convergence delay of FL and resource consumption. The challenge arises since the UAV's placement affects both sensor coverage and transmission delay, especially in inhomogeneous sensor networks. The core idea of our method is to equalize the computation and communication times of UAVs through optimal UAV placement, spectrum allocation, and power control. We provide theoretical and numerical analysis to evaluate the performance of our proposed scheme. The results show the superior performance of our approach compared to the state-of-the-art literature in terms of FL convergence delay, coverage, and power consumption.

7.2 System Model

We consider a wireless network consisting of a federated learning ground server located at $(x_s, y_s, 0)$ and a set of $\mathcal{U}$ UAVs, where $i \in \{1, ..., U\}$ with homogeneous processing capabilities. Each UAV i is located at (x_i, y_i, h_i), and has a maximum transmission power

P_i. The UAVs cover a field of sensors that are randomly distributed according to an inhomogeneous Poisson point process (IHPPP) with a known density. The exact locations of the sensors are however unknown. The sensors are homogeneous in the sense that they collect and transmit the same number of samples. The communication between the UAVs and the server is OFDM-based. There is a set $\mathcal{N}$ of N subchannels, $n \in \{1,...,N\}$.

Each UAV $i \in \mathcal{U}$ has a set of D_i data samples. By $\{\boldsymbol{x}_{il}\}_{l=1}^{D_i}$, $\boldsymbol{x}_{il} \in \mathbb{R}^d$, we represent the set of input data samples that UAV i receives from the sensors in its coverage area. Besides, $\{y_{il}\}_{l=1}^{D_i}$, $y_{il} \in \mathbb{R}$, is the corresponding set of outputs. Let $\boldsymbol{\omega}$ be the parameter vector relating the two vectors at UAV i, and $f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il})$ be the loss function for the lth data sample. The total loss function of UAV i, $F_i(\boldsymbol{\omega}, \boldsymbol{x}_{i1}, y_{i1}, \ldots, \boldsymbol{x}_{iD_i}, y_{iD_i})$, or simply $F_i(\boldsymbol{\omega})$, is

$$F_i(\boldsymbol{\omega}) \coloneqq \frac{1}{D_i} \sum_{l=1}^{D_i} f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il}). \tag{7.1}$$

The global loss function is defined as (Dinh et al., 2021)

$$F(\boldsymbol{\omega}) \coloneqq \sum_{i=1}^{U} \frac{D_i}{D} F_i(\boldsymbol{\omega}) = \frac{1}{D} \sum_{i=1}^{U} \sum_{l=1}^{D_i} f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il}), \tag{7.2}$$

where $D = \sum_{i=1}^{U} D_i$. The objective of FL is to find a shared model $\boldsymbol{\omega}^*$ that minimizes the global loss function

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega} \in \mathbb{R}^{\boldsymbol{d}}}{\arg\min} F(\boldsymbol{\omega}) \coloneqq \frac{1}{D} \sum_{i=1}^{U} \sum_{l=1}^{D_i} f_i(\boldsymbol{\omega}, \boldsymbol{x}_{il}, y_{il}). \tag{7.3}$$

We adopt the FEDL algorithm (Dinh et al., 2021) to solve the objective function in (7.3). A summary of the procedure is provided in Algorithm 7.1.

The time required for one global iteration yields

$$T_{\text{global}} = T^{\text{cm}} + T^{\text{cp}}(\kappa). \tag{7.6}$$

In (7.6), T^{cm} is the communication time between the UAVs and the server. Besides, T^{cp} is the computation time required to achieve the accuracy κ. The number of iterations for the convergence of the local model is

$$I(\kappa) = \alpha_1 \log \frac{\alpha_2}{\kappa}. \tag{7.7}$$

The constants α_1 and α_2 depend on the condition number of the Hessian matrix of $F_i(.)$ (Dinh et al., 2021).

Algorithm 7.1 The federated learning procedure (Dinh et al., 2021)

1: In the first global iteration $m = 0$, the server generates and broadcasts an initial global model $\boldsymbol{\omega}^0$ to all UAVs. It also determines the local accuracy parameter $\kappa \in (0,1)$.
2: In the following global iterations, each UAV i receives $\boldsymbol{\omega}^{m-1}$ and $\nabla \bar{F}^{m-1}$ from the server and solves $\boldsymbol{\omega}_i^m = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^d} J_i^m(\boldsymbol{\omega})$,

$$J_i^m(\boldsymbol{\omega}) \coloneqq F_i(\boldsymbol{\omega}) + \langle \eta \nabla \bar{F}^{m-1} - \nabla F_i(\boldsymbol{\omega}^{m-1}), \boldsymbol{\omega} \rangle, \tag{7.4}$$

where $\langle ., . \rangle$ is the inner product of two vectors, η is the learning rate. The resulting $\boldsymbol{\omega}_i^m$ satisfies the accuracy $\kappa \in (0,1)$, i.e.,

$$||\nabla J_i^m(\boldsymbol{\omega}_i^m)|| \leq \kappa ||\nabla J_i^m(\boldsymbol{\omega}^{m-1})||, \quad \forall i \in \mathcal{U}, \tag{7.5}$$

$\kappa = 0$ solves the local problem optimally, whereas $\kappa = 1$ means that the local model doesn't change.
3: The UAVs send their local models $\boldsymbol{\omega}_i^m$ and local gradients $\nabla F_i(\boldsymbol{\omega}_i^m)$ to the server. It aggregates them to calculate $\boldsymbol{\omega}^m = \sum_{i=1}^{U} \frac{D_i}{D} \boldsymbol{\omega}_i^m$ and $\nabla \bar{F}^m = \sum_{i=1}^{U} \frac{D_i}{D} \nabla \bar{F}_i(\omega_i^m)$. The server then transmits these values to all UAVs.
4: The global iterations, i.e., the collection of updating and transmission processes are repeated until convergence, i.e., until achieving a certain difference between the current and the minimal loss function. Formally, $F(\boldsymbol{\omega}^m) - F(\boldsymbol{\omega}^*) \leq \varepsilon$.

7.3 The Effect of Coverage and Resource Allocation on the FL Convergence Time

As described previously, we aim to maximize the network coverage while ensuring minimizing the FL convergence time and ensuring the efficiency of resource expenditure. Before proceeding to the formal problem statement, we analyze the relationship between the computation time and network coverage and the relationship between the communication time and resource allocation.

7.3.1 FL Computation Time and Network Coverage

The computation time of UAV i in each global iteration m, T_i^{cp}, is the time required to update the model using the data received from the sensors in the UAV's coverage region. Let C be the number of CPU cycles required for computing one data sample. Besides, f_{CPU} is the UAVs' CPU frequency. Then we have

$$T_i^{\text{cp}} = I \frac{CD_i}{f_{\text{CPU}}}, \quad \forall i \in \mathcal{U}, \tag{7.8}$$

where D_i is the number of data samples at UAV i and I is the number of local iterations as given in (7.7).

As a consequence of the inhomogeneous sensor distribution, the number of sensors covered by each UAV can differ significantly depending on its location. Such a difference results in long computation times for the UAVs covering dense areas and *vice versa*. In a synchronous FL scheme, the server requires the input of all UAVs for global aggregation. As such, the model update time is determined by the UAV with the highest T_i^{cp}. Since the UAVs are homogeneous, that UAV is the one that covers the highest number of sensors and, consequently, has the highest number of data samples

$$T^{\text{cp}} = \max_{i \in \mathcal{U}} T_i^{\text{cp}}. \tag{7.9}$$

7.3.2 FL Communication Time and Resource Allocation

At iteration m, the uplink data rate of UAV i is given by

$$r_{i,m}(\beta_{in,m}, p_{in,m}, x_i, y_i, h_i) = \sum_{n \in N} B^{\text{UL}} \beta_{in,m} \log\left(1 + \frac{p_{in,m}|g_{in,m}|^2}{\beta_{in,m} L_i \sigma^2}\right), \tag{7.10}$$

where B^{UL} is the uplink subchannel bandwidth and $g_{in,m}$ is the fading channel gain between UAV i and the server on subchannel n. Besides, $\beta_{in,m}$ represents the fraction of subchannel $n \in \{1,...,N\}$ allocated to UAV i such that $\sum_{n \in N} \beta_{in,m} \leq 1$. We allow only for full subchannel allocation so that $\beta_{in,m} \in \{0,1\}$. Furthermore, $p_{in,m}$ is the transmission power of UAV i on subchannel n such that $\sum_{n \in N} p_{in,m} \leq P_i$. P_i is the maximum power constraint for UAV i. The path loss L_i between UAV i and the server is determined by the air-to-ground channel model (Alzenad et al., 2017)

$$L_i^{\text{dB}}(x_i, y_i, h_i) = \frac{A}{1 + a\exp\left(-b[\frac{180}{\pi}\tan^{-1}\left(\frac{h_i}{d_i}\right) - a]\right)} + 10\log\left(d_i^2 + h_i^2\right) + Y, \tag{7.11}$$

where $A = \eta_{\text{LoS}} - \eta_{\text{NLoS}}$ is the mean loss incurred in addition to free space path loss due to shadowing and scattering for the LoS and NLos paths. Besides, a and b are constants that depend on the environment (Al-Hourani et al., 2014). $d_i = \sqrt{(x_i - x_s)^2 + (y_i - y_s)^2}$ is the horizontal distance between the ground server and UAV i. Furthermore, $Y = 20\log\left(\frac{4\pi f_c}{c}\right) + \eta_{\text{NLoS}}$. Finally, f_c is the carrier frequency and c is the speed of light.

Different UAV-server links have different lengths and qualities. Therefore, they can be significantly distinct concerning data rates and transmission time, as reflected by (7.10). Here we consider the transmission times in both the uplink, $T_{i,m}^{\text{UL}}$, and downlink, $T_{i,m}^{\text{DL}}$ (Zeng et al., 2020). Let $S_{L,i}$ be the total size of the local model and the gradient of UAV i

and let S_G be the total size of the global model and the gradients in the server, then

$$T_{i,m}^{\mathrm{UL}} = \frac{S_{\mathrm{L},i}}{r_{i,m}}, \quad T_{i,m}^{\mathrm{DL}} = \frac{S_{\mathrm{G}}}{r_{i,m}}. \tag{7.12}$$

Due to the channel reciprocity and without loss of generality, we assume that the uplink and downlink data rates are equal.

We define the FL communication time as the time required to communicate the model parameters and the gradients between the UAVs and the server. Similar to the computation time, the total time required to complete an iteration is determined by the longest transmission delay; that is,

$$T_m^{\mathrm{cm}} = \max_{i \in \mathcal{U}} \left(T_{i,m}^{\mathrm{UL}} + T_{i,m}^{\mathrm{DL}} \right). \tag{7.13}$$

7.3.3 Problem Formulation

Our objective is to minimize the time needed to complete one global iteration of FL in UAV-enabled networks.

$$\underset{\beta_{in,m}, p_{in,m}, x_i, y_i, h_i}{\text{minimize}} \quad (T^{\mathrm{cp}} + T_m^{\mathrm{cm}}) \tag{7.14}$$

subject to

$$p_{in,m} \geq 0, \quad i \in \mathcal{U}, n \in \mathcal{N}, \tag{7.15}$$

$$\sum_{n \in N} p_{in,m} \leq P_i, \quad \forall i \in \mathcal{U}, \tag{7.16}$$

$$\beta_{in,m} \in \{0,1\}, \quad i \in \mathcal{U}, n \in \mathcal{N}, \tag{7.17}$$

$$\sum_{i \in \mathcal{U}} \beta_{in,m} \leq 1, \quad n \in \mathcal{N}, \tag{7.18}$$

$$x_{\min} \leq x_i \leq x_{\max}, \quad i \in \mathcal{U}, \tag{7.19}$$

$$y_{\min} \leq y_i \leq y_{\max}, \quad i \in \mathcal{U}, \tag{7.20}$$

$$h_{\min} \leq h_i \leq h_{\max}, \quad i \in \mathcal{U}, \tag{7.21}$$

where T^{cp} and T_m^{cm} are given by (7.9) and (7.13), respectively. Constraints (7.19) and (7.20) ensure that the UAVs are within the field boundaries $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$. The constraint in (7.21) ensures that the UAV's altitude remains within the technically feasible interval $[h_{\min}, h_{\max}]$.

Solving the optimization problem (7.14) is challenging for several reasons. First, both T^{cp} and T_m^{cm} depend on the UAVs' locations. That implies that placing the UAVs to minimize the computation time and improve coverage can change the UAV-server channel status which affects the transmission time. Besides, the full (binary) allocation of OFDM subchannels renders subchannel scheduling a hard integer programming problem. Fur-

thermore, the UAVs' power constraints can vary. Hence, we solve (7.14) by decomposing it into two parts: (i) Computation time minimization through optimal UAV placement while ensuring the maximum coverage, and (ii) Communication time minimization through fair resource allocation while ensuring the optimal resource expenditure. A detailed description follows in Section 7.4 and Section 7.5, correspondingly.

7.4 Computation Time Minimization Through UAV Coverage

Based on (7.9), one can minimize the computation time of FL by shrinking the coverage area of UAVs in dense areas to reduce the number of covered sensors. While UAVs in low-density areas must increase their coverage area to maximize coverage; nevertheless, this solution is challenging to implement due to the inhomogeneous sensor distribution and the unknown sensors' locations. In this section, we propose a novel heuristic algorithm to solve this problem. Before proceeding to describe this algorithm, we explain the physical parameters that determine UAV coverage.

Every UAV $i \in \mathcal{U}$ located at (x_i, y_i, h_i) has a disk-shaped coverage region with radius R_i. Let $L_{\text{th}}^{\text{dB}}$ be the maximum allowable path loss for any sensor. Then $L_{\text{th}}^{\text{dB}}$ determines the maximum coverage radius of a UAV, denoted by $R_{\max}$. More precisely, any sensor located at a horizontal distance below $R_{\max}$ to a UAV $i \in \mathcal{U}$ experiences a path loss less than $L_{\text{th}}^{\text{dB}}$ when communicating to that UAV; In this case, we say that UAV i covers that sensor. The relation between the threshold path loss $L_{\text{th}}^{\text{dB}}$ and the maximum coverage radius $R_{\max}$ is

$$L_{\text{th}}^{\text{dB}} = \frac{A}{1 + a\exp\left(-b[\frac{180}{\pi}\theta_{\text{opt}} - a]\right)} + 20\log\left(\frac{R_{\max}}{\cos(\theta_{\text{opt}})}\right) + Y. \tag{7.22}$$

In (7.22), θ_{opt} is the optimal elevation angle, which is an environment-dependent factor that determines the optimum altitude to maximize the coverage radius (Al-Hourani et al., 2014). For example, the optimal elevation angle for an urban environment $\theta_{\text{opt}} = 42.44°$. The optimal altitude for $R_{\max}$ yields

$$h_{\text{opt}} = R_{\max}\tan(\theta_{\text{opt}}). \tag{7.23}$$

7.4.1 Problem Statement

The average number of sensors covered by UAV $i \in \mathcal{U}$ is a function of the UAV's location. Formally,

$$\bar{n}_i = \int\int_{A_i} \lambda(x,y)dxdy, \quad \forall i \in \mathcal{U}, \tag{7.24}$$

where $\lambda(x,y)$ is the IHPPP density function and A_i is UAV i's coverage area. Thus, even for UAVs with the same coverage radius, the value of $\bar{n}_i$, thereby the computation time, can differ drastically. Our objective is to maximize the total number of sensor covered by all UAVs while keeping $\bar{n}_i$ bounded between $[N_{\min}, N_{\max}]$. The values of $N_{\min}$ and $N_{\max}$ depend on the sensor density, number of UAVs, and the UAVs' computation capabilities. The upper bound prevents long computation times, whereas the lower bound determines the coverage; hence the difference $N_{\max} - N_{\min}$ balances the trade-off between maximizing coverage and equalizing the UAVs' computation delays. The problem is formally stated as

$$\underset{x_i,y_i,h_i}{\text{maximize}} \sum_{i\in\mathcal{U}} \bar{n}_i(x_i,y_i,h_i), \tag{7.25}$$

subject to

$$(7.19),(7.20),(7.21), \tag{7.26}$$

$$N_{\min} \le \bar{n}_i \le N_{\max}, \quad \forall i \in \mathcal{U}, \tag{7.27}$$

$$\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \ge R_i + R_j \quad \forall i,j \in \mathcal{U}, i \ne j. \tag{7.28}$$

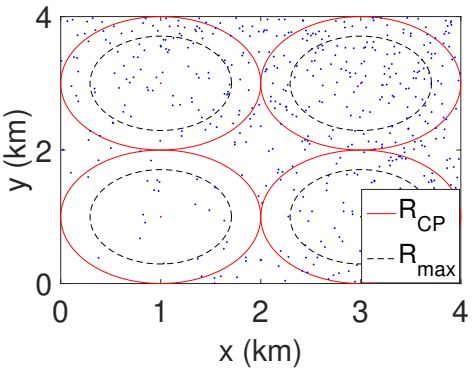
Constraint (7.28) is to avoid an overlap between the UAVs' coverage regions.

7.4.2 Solution

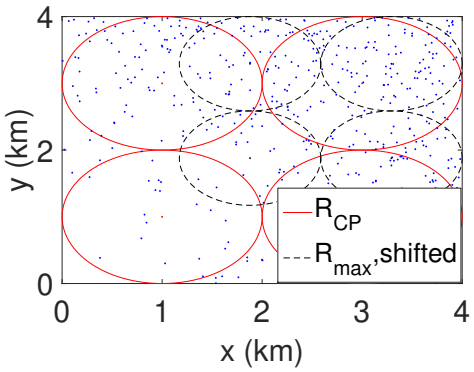
To solve (7.25), we propose the algorithm described below.
Step 1: The algorithm determines the UAVs' initial placement $(x_{i,\mathrm{CP}}, y_{i,\mathrm{CP}})$ and coverage radius R_{CP} using the well-known *circle packing* (CP) method (Szabó et al., 2005). The CP method finds the locations and the maximum radius of U equal circles that one can pack in a surface to maximize the packing density without overlapping. Table 7.1 shows the maximum radius as a function of the square length E for different number of UAVs.
Step 2: Sensors located outside $R_{\max}$ experience high path loss and are not covered by any UAV. Therefore, the UAVs' radii are adjusted to $R_0 = \min\{R_{\mathrm{CP}}, R_{\max}\}$.
Step 3: The locations of the UAVs depend on R_0 as follows:

- If $R_0 = R_{\mathrm{CP}}$, the UAVs remain at the locations determined by CP, i.e., $(x_{i,\mathrm{CP}}, y_{i,\mathrm{CP}})$. Besides, $h_0 = R_0 \tan(\theta_{\mathrm{opt}})$.

- If $R_0 = R_{\max}$, some gaps will result from reducing the radius from R_{CP} to $R_{\max}$. Therefore, we shift the UAVs in the direction of increasing density in proportion to $(R_{\mathrm{CP}} - R_{\max})$ to increase coverage. Besides, $h_0 = R_0 \tan(\theta_{\mathrm{opt}})$. This case is illustrated in Fig. 7.1 for a scenario with four UAVs. Here, we select $\lambda(x,y) = 3(x^2+y^2)$, $R_{\mathrm{CP}} = 1$ km and $R_{\max} = 0.707$ km. If located at the position determined by CP, the average number of sensors that the UAVs cover is $\bar{N}_{\mathrm{total}} = 193.23$, as



(a) Before shifting the UAVs $\bar{N}_{\text{total}} = 193.23$ nodes.



(b) After shifting the UAVs $\bar{N}_{\text{total}} = 275.97$ nodes.

Figure 7.1: Illustration of the effect of shifting the UAVs.

shown in Fig. 7.1a. Shifting the UAVs in the direction of higher density, as shown in Fig. 7.1b, improves the coverage so that $\bar{N}_{\text{total}} = 275.97$.

Step 4: Up to this point, all UAVs have the same radius R_0 and altitude h_0 but different horizontal positions $(x_i, y_i), \forall i \in \mathcal{U}$. In this step, the UAVs' radii R_i are adjusted according to the number of sensors covered by each UAV. First, the algorithm determines the UAV with the highest $\bar{n}_i$ using (7.24), and denotes this UAV by $D^{(1)}$, where $D^{(j)}$ is the UAV with the jth highest average number of sensors. Then, the following two cases are considered:

- If $\bar{n}_{D^{(j)}} > N_{\max}$, the algorithm reduces the altitude $h_{D^{(j)}}$ by a constant Δ_h, with $h_{D^{(j)}} - \Delta_h \geq h_{\min}$. Consequently, $R_{D^{(j)}}$ reduces by Δ_R. Then it shifts UAV $D^{(j)}$ in proportion to Δ_R in the direction of higher node density. The procedure continues until $N_{\min} \leq \bar{n}_{D^{(j)}} \leq N_{\max}$. Afterwards, the algorithm shifts the remaining UAVs in the direction of increasing sensor density without overlapping.

- If $\bar{n}_{D^{(j)}} < N_{\min}$, the algorithm increases $h_{D^{(j)}}$ by a constant Δ_h, with $h_{D^{(j)}} + \Delta_h \leq h_{\max}$. Thus, $R_{D^{(j)}}$ increases by Δ_R, subject to the constraint $R_{D^{(j)}} + \Delta_R \leq R_{\max}$. Then it shifts UAV $D^{(j)}$ in proportion to Δ_R in the direction of decreasing sensor density. This continues until $N_{\min} \leq \bar{n}_{D^{(j)}} \leq N_{\max}$. Finally, the algorithm shifts the remaining UAVs in the direction of the decreasing sensor density without overlapping.

Step 5: The algorithm recalculates $\bar{n}_i, \forall i \in \mathcal{U} - \{D^{(1)}, \ldots, D^{(j)}\}$. The same procedure is repeated for the remaining UAVs $D^{(j+1)}, D^{(j+2)}, \ldots, D^{(U)}$. We summarize the procedure in Algorithm 7.2.

7.4.3 Complexity Analysis

The CP problem is intractable, as there is no general method for packing U circles in a square. However, the solution for CP is listed in tables for a large number of circles, it

Number of UAVs U	Coverage radius R_{CP}	Total coverage (%)
2	$\frac{1}{2+\sqrt{2}}E$	53.90
3	$\frac{2}{4+\sqrt{2}+\sqrt{6}}E$	60.96
4	$\frac{1}{4}E$	78.54
5	$\frac{1}{2}(\sqrt{2}-1)E$	67.38

Table 7.1: Equal CP in a square of edge length E

is optimal in a few cases and approximate in the rest (Specht, 2022). Therefore, the CP part renders a simple value assignment operation. The remaining part of the algorithm requires solving $\frac{U(U-1)}{2}$ non-linear problems with non-convex constraints. An approximate solution is found by dividing the area into convex sub-regions.

7.5 Communication Time Minimization Through Fair Resource Allocation

The lengths and quality of the UAV-to-server communication links can vary significantly between the UAVs; thus, the communication times of the UAVs become crucially different. To reduce this difference, we implement a joint subchannel scheduling and power control algorithm to maintain fairness among the UAVs that participate in the learning process while retaining the efficiency of resource expenditure. At each round m and for each UAV $i \in \mathcal{U}$, we define a utility function u_i as

$$u_i(W_{i,m}) = \begin{cases} \frac{1}{\alpha}(W_{i,m})^{\alpha} & \alpha \leq 1, \alpha \neq 0 \\ \log(W_{i,m}) & \alpha = 0 \end{cases}. \tag{7.29}$$

The utility function in (7.29) depends on the average throughput so far, $W_{i,m}$, and α is the fairness parameter. Namely, $\alpha = 0$ gives the proportionality fair (PF) rule whereas $\alpha = 1$ results in a maximum throughput scheduling.
To guarantee fairness, we use gradient-based scheduling to find the uplink data rate $\boldsymbol{r}^*_m = \{r^*_{1,m}, ..., r^*_{U,m}\}$ that maximizes the weighted sum of the UAVs' data rates, where the weights are the gradients of the UAVs' utility functions. Formally,

$$\underset{\beta_{in,m}, p_{in,m}}{\text{maximize}} \sum_{i \in U} \frac{\partial u_i(W_{i,m})}{\partial W_{i,m}} r_{i,m} = \underset{\beta_{in,m}, p_{in,m}}{\text{maximize}} \sum_{i \in U} (W_{i,m})^{\alpha-1} r_{i,m}. \tag{7.30}$$

Algorithm 7.2 Joint coverage maximization and computation time minimization

Input: U, E, $\lambda(x,y)$, L_{th}, θ_{opt}, $N_{\max}$, $N_{\min}$.
Output: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}$.
Find the maximum coverage radius $R_{\max}$ using (7.22).
Find the CP radius R_{CP} using Table 7.1.
$\forall i \in \mathcal{U}$ find the initial placement (x_i, y_i) using the circle packing theory.
if $R_{\max} \geq R_{\text{CP}}$ **then**
    $R_i = R_{\text{CP}}$. The UAVs' locations do not change.
else
    $R_i = R_{\max}$.
    Shift the UAVs in the direction of the higher node density without overlapping.
end if
$h_i = R_i \tan(\theta_{\text{opt}})$.
for $j = 1 : U$ **do**
    Find the UAV with the jth highest average number of sensors, $D^{(j)}$.
    if $\bar{n}_{D^{(j)}} > N_{\max}$ **then**
        while $\bar{n}_{D^{(j)}} > N_{\max}$ **do**
            Reduce $R_{D^{(j)}}$ by Δ_R by reducing $h_{D^{(j)}}$ by Δ_h.
            Shift UAV $D^{(j)}$ towards higher node density in proportion to Δ_R.
            Update the location of UAV $D^{(j)}$, $(x_{D^{(j)}}, y_{D^{(j)}})$.
            Find $\bar{n}_{D^{(j)}}$ using (7.24).
        end while
        Shift the remaining UAVs in the direction of higher node density without overlapping.
    else if $\bar{n}_{D^{(j)}} < N_{\min}$ **then**
        while $\bar{n}_{D^{(j)}} < N_{\min}$ **do**
            Increase $R_{D^{(j)}}$ by Δ_R by increasing $h_{D^{(j)}}$ by Δ_h such that $R_{D^{(j)}} \leq R_{\max}$.
            Shift UAV $D^{(j)}$ towards its initial position in proportion to Δ_R.
            Update the location of UAV $D^{(j)}$, $(x_{D^{(j)}}, y_{D^{(j)}})$.
            Find $\bar{n}_{D^{(j)}}$ using (7.24).
        end while
        Shift the remaining UAVs in towards their initial positions without overlapping.
    end if
end for

Substituting the value of $r_{i,m}$ from (7.10) gives:

$$\underset{\beta_{in,m}, p_{in,m}}{\text{maximize}} \sum_{i \in U} (W_{i,m})^{\alpha-1} \sum_{n \in N} B^{\text{UL}} \beta_{in,m} \log\left(1 + \frac{p_{in,m}|g_{in,m}|^2}{\beta_{in,m} L_i \sigma^2}\right), \tag{7.31}$$

subject to (7.15), (7.16), (7.17), and (7.18).

We adopt the approach in (Huang et al., 2009) to solve problem (7.31) for the uplink. The method starts with a subchannel allocation phase, where each subchannel is allocated to one UAV at most. Then, each UAV uses the water-filling concept to allocate the power to the subchannels assigned to it, subject to its power constraint. As a result,

$$p^*_{in,m} = \min\left(\left(\frac{\beta^*_{in,m}}{\nu_{i,m}} - \frac{1}{c_{in,m}}\right)^+, \frac{s_{in}}{e_{in,m}}\right), \tag{7.32}$$

where $c_{in,m} = \frac{|g_{in,m}|^2}{L_i \sigma^2}$. Besides, $\nu_{i,m}, \forall i \in \mathcal{U}$ satisfies the UAVs' power constraint $\sum_{n \in N} p^*_{in,m} = P_i$.

The power allocation scheme given by ((7.32)) corresponds to a constraint on the maximum SNR on subchannel n for UAV i, s_{in}, which ensures the max-min fairness. In other words, the UAV with the weakest channel uses its maximum available power, whereas other UAVs reduce their power consumption to save energy. Formally, it aims at

$$\underset{p_{in,m}}{\text{maximize}} \left(\underset{i \in \mathcal{U}}{\text{minimum}} \;\; r_{i,m}\right) \tag{7.33}$$

subject to constraints (7.15) and (7.16).

7.5.1 Complexity Analysis

The computational complexity of subchannel allocation is $\mathcal{O}(NU + N \log N)$ and the complexity of the water filling algorithm is $\mathcal{O}(NU)$ (Huang et al., 2009), resulting in a total complexity of $\mathcal{O}(NU + N \log N)$ for resource allocation.

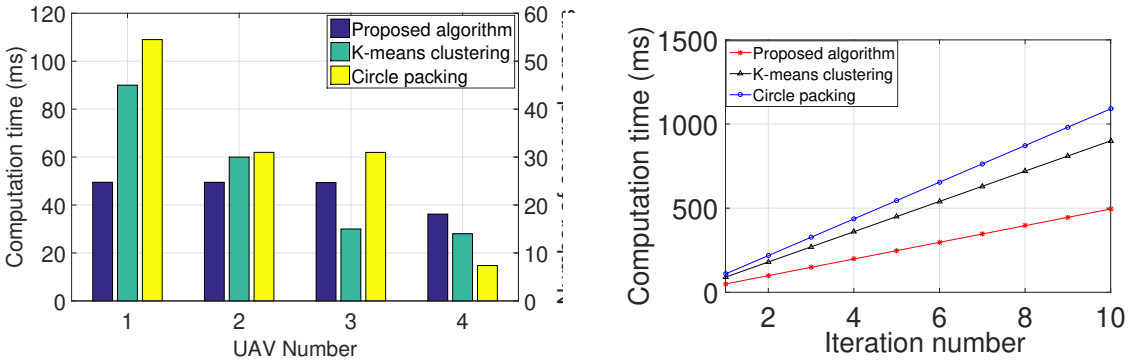
7.6 Numerical Results

In this section we evaluate the proposed algorithm's FL convergence time and coverage probability. We consider a 2 km $\times$ 2 km square region with IHPPP distributed sensors with density function $\lambda(x,y) = 15(x^2+y^2)$ sensor/km^2 and four UAVs. Besides, we select $\eta_{LoS} = 1$ dB, $\eta_{NLoS} = 20$ dB, $a = 9.61$, $b = 0.61$, $f_c = 2$ GHz, and $c = 3 \times 10^8$ m/s. For each UAV, the CPU frequency is $f_{\text{CPU}} = 2$ GHz and $C = 20$ cycles/bit. The size of a data sample collected from each sensor is 200 Kb. The training size D_i at UAV i in bits is the product of $\bar{n}_i$ and the data size of each sensor. There are 20 OFDM subchannels, each with a bandwidth $B^{\text{UL}} = 1$ MHz. The maximum power constraint is 1 W for all UAVs. The channel gains follow the Ricean distribution. The path loss is obtained by substituting the UAVs' locations into (7.11). We select $\alpha = 0$ for proportionally fair resource allocation.

First, we evaluate the FL computation time for our algorithm and compare it to the circle packing and also the k-means clustering (Sun and Masouros, 2018) methods. The k-

means clustering method aims at maximizing the sensor coverage by dividing the sensors into U clusters. It then finds the maximum UAV coverage radius and the placement that fits into these subareas to maximize coverage. According to Table 7.1, we have $R_{\text{CP}} = 0.5$ km. By (7.22), the maximum coverage radius that corresponds to $L_{th} = 100$ dB is equal to $R_{\max} = 0.707$ km. Since $R_{\text{CP}} \leq R_{\max}$, the initial radius and position of the UAVs in our proposed algorithm are equivalent to the CP method.

Fig. 7.2a shows the computation time for one local iteration at each of the four UAVs, for the three coverage methods. In CP, the four UAVs have equal radius R_{CP}, but $\bar{n}_i$ varies drastically between them. This is due to the inhomogeneous nodes distribution that causes a significant difference in the computation time, thereby slowing down the FL process. The same holds for the k-means method. In contrast, our proposed scheme guarantees that $\bar{n}_i \in [18, 25]$; therefore, the computation times of the UAVs differ only slightly. Most importantly, the maximum delay, which affects the FL, decreases. Fig. 7.2b shows the elapsed time in updating the local model in each iteration. We show the UAV with the highest computation delay as it determines the update time in synchronous FL.



(a) The computation time for one local iteration for the four UAVs.

(b) Cumulative computation time in one global iteration.

Figure 7.2: FL computation time

Fig. 7.3 shows the coverage probability for the three UAV placement methods. Naturally, the proposed algorithm minimizes the computation delay at the expense of reducing the coverage probability. Its performance can be improved by increasing $N_{\max}$; nevertheless, this increases T^{cp}. It should be noted that the k-means method, despite the highest coverage probability, results in high computation delay. Besides, it is extremely costly in terms of power consumption, as the UAVs' radii are maximized. Indeed, the UAVs are located to maximize the coverage even if a significant part of the UAVs' coverage region lies outside the square field.

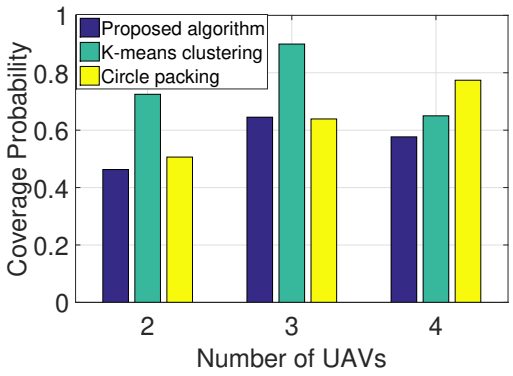
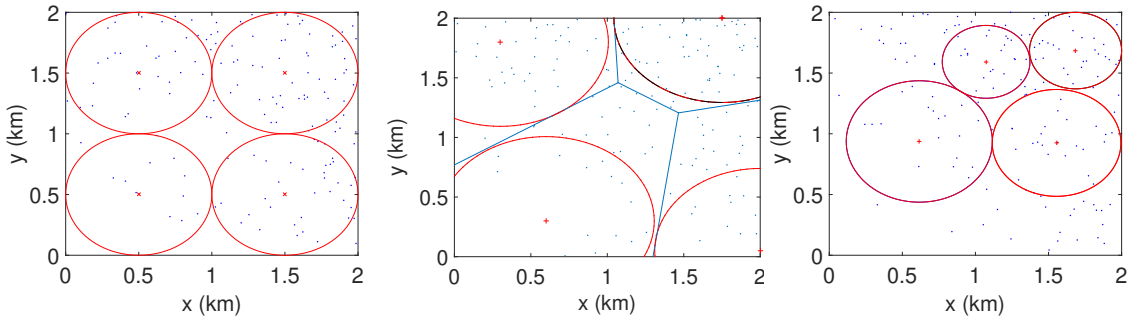


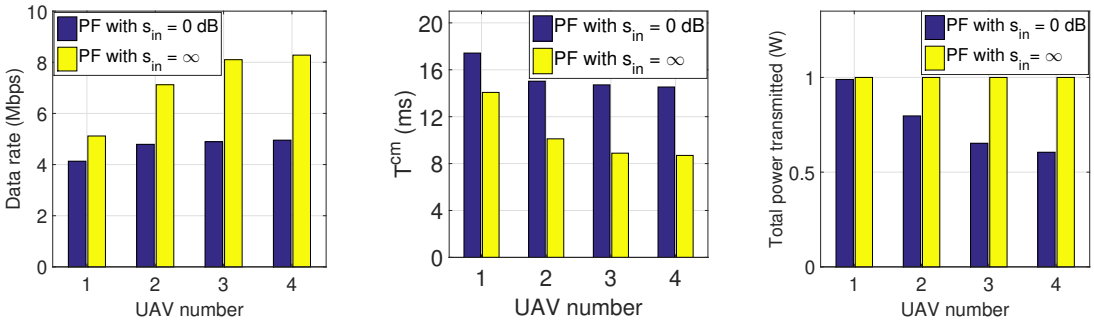
Figure 7.3: Coverage probability for different number of UAVs.

After the UAV placement as shown in Fig. 7.4c, the resource allocation procedure starts.



(a) Circle packing (b) K-means clustering (c) Proposed algorithm

Figure 7.4: Three UAV placement methods.



(a) Data rates (b) Communication time (c) Transmission power

Figure 7.5: Resource allocation.

Fig. 7.5a shows the data rate between the UAVs and the server in one global iteration, averaged over 1000 rounds. From Fig. 7.5b, the proposed method results in homogeneous data rates among the UAVs, and thereby homogeneous communication times. The latter figure shows the sum of the communication times of the local data of size $S_{L,i} = 36\,\text{kbit}, \forall i \in \mathcal{U}$ and the global data of size $S_G = 36\,\text{kbit}$. The homogeneous data rates and communication times are achieved by maximizing the power allocated to the UAV with the minimum data rate as shown in Fig. 7.5c.

Part III

Conclusion & Future Work

Chapter 8

Conclusion & Future Work

8.1 Conclusion

This thesis studies different approaches to resource allocation and performance optimization in MEC under different scenarios. Since wireless networks are characterized by high uncertainty and incomplete information, most of the proposed solution approaches are based on MAB algorithms. The only exception is presented in Chapter 7, which introduces a heuristic method for optimizing the performance of a FL process running on UAVs. This heuristic approach is replaced by a MAB-based solution in Chapter 6, due to the lack of network information.

Chapter 4 addresses the service placement problem under unknown service demand in a dynamic environment by proposing a distributed and adaptive multi-agent BAI algorithm in linear bandits for the fixed confidence setting. SBSs equipped with edge servers act as agents, and services are modeled as arms, with demand represented as a linear function of contextual information. To reduce communication overhead, each SBS initiates communication only when a significant change in the observed information is detected. Numerical results show that the proposed algorithm effectively identifies the optimal service to place at SBSs, rather than in the cloud, to maximize the reduction in the total user delay. Furthermore, collaboration among SBSs reduces the sample complexity per SBS (agent) by up to M times, where M is the number of SBSs. We also evaluate the algorithm's performance on synthetic data and show the significant improvements in the sample complexity compared to static and semi-adaptive alternatives. Finally, we derive upper bounds on both sample complexity and communication cost and analyze the algorithm robustness to communication failures.

Chapter 5 addresses the problem of decentralized task offloading and load balancing in dense networks, where users aim to offload their tasks to servers capable of completing them before an expiry time while achieving a desired load distribution. After modeling the problem as a mean-field MAB game, we prove that the system converges to a steady-state population profile theoretically and validate the result through numerical simulation. To guide the system toward a desired user distribution across servers, we propose an algorithm that adjusts the users' perceived rewards. Both theoretical analysis and simulations

show that the method converges to the target distribution. We show that the algorithm performs well across various user densities, with smoother convergence observed at higher densities due to a better approximation of the environment by the mean field model.

Chapter 6 considers a scenario in which UAVs collect data from ground sensors and act as FL clients. The goal is to jointly maximize UAV coverage and minimize FL delay by minimizing the maximum delay among the UAVs. These two objectives are conflicting. Coverage is controlled by adjusting UAVs' beamwidths, but user distribution and locations are unknown. To address this, the problem is modeled as a multi-objective MAB, where the FL server acts as the agent and each arm corresponds to a specific beamwidth configuration of the UAVs, determining both the amount of collected data and the resulting FL update time. We first apply Pareto UCB1 and scalarization-based approaches to solve the problem and show that Pareto-UCB1 identifies all optimal arms and plays them fairly. In contrast, linear scalarization fails to identify some optimal arms, leading to higher regret and unfairness compared to Chebyshev scalarization. Then, we propose a scalarization-based generalized multi-objective MAB algorithm that defines optimal arms as those that maximize the ratio of expected reward to the expected energy cost, providing an energy-efficient solution. We demonstrate the effectiveness of the proposed algorithm in identifying the optimal arm across various special cases and compare their regret and fairness, and analyze its error probability.

Chapter 7 explores the impact of UAV deployment on FL performance, where UAVs serve as FL clients using data aggregated from ground sensors. Assuming a known sensor distribution but unknown sensor locations, the objective is to minimize the local update time, which depends on both communication delay and sensor coverage. The problem is addressed using a two-phase approach: a heuristic method for sensor coverage, followed by a fair resource allocation phase that includes subchannel and power assignment. Numerical results show that the proposed method effectively reduces straggler delays by achieving more uniform coverage across UAVs, leading to balanced computation times. The resource allocation strategy further equalizes communication delays. While the approach may not always yield the highest total coverage, it outperforms baseline methods such as circle packing and K-means clustering in terms of FL update delay, while still ensuring certain coverage.

8.2 Future Work

There are several promising directions for future research, which can broadly be categorized into two areas: developing the algorithms and practical implementation.

8.2.1 Algorithm Development

Starting with the adaptive BAI algorithm in the fixed-confidence setting of linear bandits, several extensions are possible. First, the current approach may be generalized to heterogeneous agents, where the set of arms differs across agents, or to dynamic environments in which the set of available arms changes over time. Another direction is to consider personalized learning, where the model parameters vary across agents. Additionally, one could extend the problem to top-m arm identification, where agents aim to identify the best subset of m arms rather than the single best. These ideas lead to the development of multi-agent extensions, where collaborative learning strategies can help reduce the sample complexity per agent. For all these variants, it is essential to derive theoretical guarantees on sample complexity and, in multi-agent scenarios, on the communication cost as well.

In the context of multi-objective MABs, the current formulation assumes that the objectives are independent. However, in many practical settings, including ours, the objectives are correlated. Learning these correlations can enhance the learning efficiency. Future work could investigate alternative definitions of the best arm, such as prioritizing some objectives, or defining the optimal arm differently such as meeting certain constraints or thresholds on different objectives. These formulations can be extended to budgeted or cost-aware settings. Furthermore, the area of multi-agent, multi-objective MABs remains under-explored (Rădulescu et al., 2020) with many open problems related to utility function design and agent interactions.

Regarding the mean-field bandit game, this work considered a simplified system model to ensure convergence to the mean-field steady state. An important extension would be to incorporate the impact of network interference into the system model and to define agent types in a way that still guarantees convergence to the MFSS. Additionally, the target population profile is assumed to be given. However, in practice, it may depend on external factors such as pricing policies or energy costs at edge servers. This motivates developing approaches to calculate the target population profile or developing incentive mechanisms to dynamically influence agent behavior. Finally, future work could study different functions or approaches for achieving the desired population profile and analyze the convergence of the proposed function.

Finally, in the problem of UAV deployment for FL, the current heuristic-based coverage strategy could be replaced with a more theoretically grounded approach.

8.2.2 Applications

The algorithms proposed in this thesis have been applied to specific problems in MEC, motivated by our research interests. However, they are more broadly applicable and can be extended to a wide range of problems both within and beyond MEC.

To the best of our knowledge, this work is the first to use the BAI setting in linear

bandits for problems in communication networks, particularly in MEC. While our study has focused on the service placement problem, many practical settings involve contextual information that can be exploited to improve learning efficiency. Both the single-agent and the proposed multi-agent BAI frameworks are well-suited to a variety of applications. For example, it can be used in content caching, where arms correspond to contents, and features reflect content popularity, user profiles, or spatial distribution. Also, in client selection in FL, where clients are selected based on features such as data quality and computational resources. Additionally, in channel scheduling, where scheduling decisions depend on features of users or channels such as queue lengths or interference levels.

In addition, mean-field MAB games can be incorporated in networks involving large populations of agents such as dense networks and device-to-device (D2D) communication. In such settings, the actions of each agent are influenced by the aggregate behavior of the population. These systems have been addressed using the mean field theory Banez et al. (2021), but these problems can be modeled using mean-field MAB to address the network uncertainty.

Another promising direction is considering MO-MAB algorithms. While much of the existing literature focuses on optimizing a single objective, such as minimizing delay, energy consumption, or maximizing throughput, real-world systems often involve multiple, often conflicting objectives. For instance, in FL, MO-MAB can be used to select clients that jointly balance convergence speed, model accuracy, and energy efficiency. Similarly, in service placement, objectives may include maximizing cache hit rate while minimizing delay or cost. In channel allocation, the framework can support optimization of throughput and fairness simultaneously.

The proposed approaches and their potential extensions represent a promising contribution to the development of more intelligent and adaptive networks, aligning with the vision for next-generation communication systems.

Bibliography

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Proceedings Systems*, 24, 2011.

Akram Al-Hourani, Sithamparanathan Kandeepan, and Simon Lardner. Optimal LAP altitude for maximum coverage. *IEEE Wireless Communications Letters*, 3(6):569–572, 2014.

Mohamed Alzenad, Amr El-Keyi, Faraj Lagum, and Halim Yanikomeroglu. 3-D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage. *IEEE Wireless Communications Letters*, 6(4):434–437, 2017.

Sanae Amani, Tor Lattimore, András György, and Lin Yang. Distributed contextual linear bandits with minimax optimal communication cost. In *International Conference on Machine Learning (ICML)*, pages 691–717. PMLR, 2023.

Barry C Arnold, Narayanaswamy Balakrishnan, and Haikady Navada Nagaraja. *A First Course in Order Statistics*. SIAM, 2008.

Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *The 23rd Annual Conference on Learning Theory (COLT)*, pages 13–p, 2010.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Javad Azizi, Branislav Kveton, Mohammad Ghavamzadeh, and Sumeet Katariya. Meta-learning for simple regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6709–6717, 2023.

Constantine A. Balanis. *Antenna Theory: Analysis and Design*. John Wiley & Sons, 4th edition, 2015.

Amir Rezaei Balef and Setareh Maghsudi. Piecewise-stationary multi-objective multi-armed bandit with application to joint communications and sensing. *IEEE Wireless Communications Letters*, 12(5):809–813, 2023.

Reginald A Banez, Lixin Li, Chungang Yang, and Zhu Han. *Mean Field Game and Its Applications in Wireless Networks*. Springer, 2021.

Daniel Becking, Heiner Kirchhoffer, Gerhard Tech, Paul Haase, Karsten Müller, Heiko Schwarz, and Wojciech Samek. Adaptive differential filters for fast and communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3367–3376, 2022.

Bouziane Brik, Adlen Ksentini, and Maha Bouaziz. Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems. *IEEE Access*, 8:53841–53849, 2020.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 23–37. Springer, 2009.

Yuwen Cao, Setareh Maghsudi, and Tomoaki Ohtsuki. Mobility-aware routing and caching: A federated learning assisted approach. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2021.

Lixing Chen and Jie Xu. Budget-constrained edge service provisioning with demand estimation via bandit learning. *IEEE Journal on Selected Areas in Communications*, 37(10):2364–2376, 2019.

Lixing Chen, Jie Xu, Shaolei Ren, and Pan Zhou. Spatio–temporal edge service placement: A bandit learning approach. *IEEE Transactions on Wireless Communications*, 17(12):8388–8401, 2018.

Mingzhe Chen, H Vincent Poor, Walid Saad, and Shuguang Cui. Convergence time optimization for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 20(4):2457–2471, 2020.

Xin Chen, Zhiyong Liu, Ying Chen, and Zhuo Li. Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks. *IEEE Access*, 7:184172–184182, 2019.

Yiming Chen, Xingyuan Hu, Shimin Gong, Zhou Su, and Bo Gu. Multi-time scale service caching and pricing in MEC systems with dynamic program popularity. *IEEE Internet of Things Journal*, 2025.

Zhirui Chen, PN Karthik, Vincent YF Tan, and Yeow Meng Chee. Federated best arm identification with heterogeneous clients. *IEEE Transactions on Information Theory*, 70(6):4258–4279, 2023.

Xiaotong Cheng and Setareh Maghsudi. Distributed task management in fog computing: A socially concave bandit game. *IEEE Transactions Green Communications and Networks*, 7(3):1486–1500, 2023.

Yae Jee Cho, Samarth Gupta, Gauri Joshi, and Osman Yağan. Bandit-based communication-efficient client selection strategies for federated learning. In *The 54th Asilomar Conference on Signals, Systems, and Computers*, pages 1066–1069. IEEE, 2020.

Alok Choudhury, Manojit Ghose, Akhirul Islam, et al. Machine learning-based computation offloading in multi-access edge computing: A survey. *Journal of Systems Architecture*, 148:103090, 2024.

Qimei Cui, Xiaohu You, Ni Wei, et al. Overview of AI and communication for 6G network: Fundamentals, challenges, and future research opportunities. *Science China Information Sciences*, 68(7):171301, 2025.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: An introduction. In *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pages 3–34. Springer, 2011.

Domenico Di Sivo, Palma Errico, and Salvatore Venticinque. Federated learning in agents based cyber-physical systems. In *Artificial Intelligence Techniques for Analysing Sensitive Data in Medical Cyber-Physical Systems: System Protection and Data Analysis*, pages 73–94. Springer, 2025.

Canh T Dinh, Nguyen H Tran, Minh NH Nguyen, Choong Seon Hong, Wei Bao, Albert Y Zomaya, and Vincent Gramoli. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking*, 29 (1):398–409, 2021.

Madalina M. Drugan and Ann Nowe. Designing multi-objective multi-armed bandits algorithms: A study. In *The International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2013.

Madalina M Drugan and Ann Nowé. Scalarization based Pareto optimal set of arms identification algorithms. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2690–2697. IEEE, 2014.

Mani Bushan Dsouza. A survey of Hadoop MapReduce scheduling algorithms. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(7), 2015.

Qiang Duan, Jun Huang, Shijing Hu, Ruijun Deng, Zhihui Lu, and Shui Yu. Combining federated learning and edge computing toward ubiquitous intelligence in 6G network: Challenges, recent advances, and future directions. *IEEE Communications Surveys & Tutorials*, 25(4):2892–2950, 2023.

Ericsson. Ericsson mobility report. https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2024, 2024. Accessed: Jan. 2025.

Ericsson. Co-creating a cyber-physical world. Technical Report GFTL-24:000856 Uen, Ericsson, July 2024. White Paper URL: https://www.ericsson.com/4a32a8/assets/local/reports-papers/white-papers/2024/co-creating-a-cyber-physical-world.pdf.

Mohammad Esmaeil Esmaeili, Ahmad Khonsari, Vahid Sohrabi, and Aresh Dadlani. Reinforcement learning-based dynamic load balancing in edge computing networks. *Computer Communications*, 222:188–197, 2024.

Vajiheh Farhadi, Fidan Mehmeti, Ting He, Thomas F La Porta, Hana Khamfroush, Shiqiang Wang, Kevin S Chan, and Konstantinos Poularakis. Service placement and request scheduling for data-intensive applications in edge clouds. *IEEE/ACM Transactions on Networking*, 29(2):779–792, 2021.

European Commission: Directorate-General for Research and Innovation. ERA industrial technologies roadmap on human-centric research and innovation for the manufacturing sector. *Publications Office of the European Union*, 2024.

Othmane Friha, Mohamed Amine Ferrag, Lei Shu, Leandros Maglaras, and Xiaochan Wang. Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies. *IEEE/CAA Journal of Automatica Sinica*, 8(4):718–752, 2021.

Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027. PMLR, 2016.

Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 88:50–71, 2017. ISSN 1084-8045.

Saeed Ghoorchian and Setareh Maghsudi. Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):279–293, 2021.

Marco Grossi. Energy harvesting strategies for wireless sensor networks and mobile devices: A review. *Electronics*, 10(6):661, 2021.

Ramki Gummadi, Ramesh Johari, Sven Schmit, and Jia Yuan Yu. Mean field analysis of multi-armed bandit games. In *Proceedings SSRN*, pages 1–28, 2013. [Online]. Available: https://ssrn.com/abstract=2045842.

Mian Guo, Mithun Mukherjee, Jaime Lloret, Lei Li, Quansheng Guan, and Fei Ji. Joint computation offloading and parallel scheduling to maximize delay-guarantee in cooperative MEC systems. *Digital Communications and Networks*, 10(3):693–705, 2024.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Haiyun He, Shuowen Zhang, Yong Zeng, and Rui Zhang. Joint altitude and beamwidth optimization for UAV-enabled multiuser communications. *IEEE Communications Letters*, 22(2):344–347, 2017.

Jiafan He, Tianhao Wang, Yifei Min, and Quanquan Gu. A simple and provably efficient algorithm for asynchronous federated contextual linear bandits. *Advances in Neural Information Processing Systems*, 35:4762–4775, 2022.

Wen He, Dazhi He, Yihang Huang, Yizhe Zhang, Yin Xu, Guan Yun-feng, and Wenjun Zhang. Bandit learning-based service placement and resource allocation for mobile edge computing. In *IEEE 31st International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–6. IEEE, 2020.

Matthew Hoffman, Bobak Shahriari, and Nando Freitas. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *Artificial Intelligence and Statistics (AISTATS)*, pages 365–374. PMLR, 2014.

Jianwei Huang, Vijay G Subramanian, Rajeev Agrawal, and Randall Berry. Joint scheduling and resource allocation in uplink OFDM systems for broadband wireless access networks. *IEEE Journal on Selected Areas in Communications*, 27(2):226–234, 2009.

Ruiquan Huang, Weiqiang Wu, Jing Yang, and Cong Shen. Federated linear contextual bandits. *Advances in Neural Information Processing Systems*, 34:27057–27068, 2021a.

Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y. Zomaya. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1552–1564, 2021b.

Yassir Jedra and Alexandre Proutiere. Optimal best-arm identification in linear bandits. *Advances in Neural Information Processing Systems*, 33:10007–10017, 2020.

Marc Jourdan and Rémy Degenne. Choosing answers in epsilon-best-answer identification for linear bandits. In *International Conference on Machine Learning*, pages 10384–10430. PMLR, 2022.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246. PMLR, 2013.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17(1):1–42, 2016.

Muhammad Asif Khan, Emna Baccour, Zina Chkirbene, Aiman Erbad, Ridha Hamila, Mounir Hamdi, and Moncef Gabbouj. A survey on mobile edge computing for video streaming: Opportunities and challenges. *IEEE Access*, 10:120514–120550, 2022.

Xunqiang Lan, Xiao Tang, Ruonan Zhang, Wensheng Lin, and Zhu Han. UAV-assisted computation offloading toward energy-efficient blockchain operations in internet of things. *IEEE Wireless Communications Letters*, 12(8):1469–1473, 2023.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Ruijie Li, Li Zhu, Guoping Zhang, Hongbo Xu, and Yun Chen. Federated learning via over-the-air computation in IRS-assisted UAV communications. *Scientific Reports*, 13 (1):8009, 2023.

Yangqianhang Li, Li Li, and Zhaorong Zhou. Joint edge caching and computation offloading for heterogeneous tasks in MEC-enabled vehicular networks. *Vehicular Communications*, 50:100860, 2024.

Chubo Liu, Kenli Li, and Keqin Li. A game approach to multi-servers load balancing with load-dependent server availability consideration. *IEEE Transactions on Cloud Computing*, 9(01):1–13, Jan. 2021. ISSN 2168-7161.

Jingyuan Liu, Zheng Chang, Kai Wang, Zhiwei Zhao, and Timo Hämäläinen. Energy-efficient and privacy-preserved incentive mechanism for mobile edge computing-assisted federated learning in healthcare system. *IEEE Transactions on Network and Service Management*, 21(4):4801–4815, 2024.

Setareh Maghsudi and Ekram Hossain. Multi-armed bandits with application to 5G small cells. *IEEE Wireless Communications*, 23(3):64–73, 2016.

Setareh Maghsudi and Ekram Hossain. Distributed user association in energy harvesting dense small cell networks: A mean-field multi-armed bandit approach. *IEEE Access*, 5:3513–3523, 2017.

Hadi Tabatabaee Malazi, Saqib Rasool Chaudhry, Aqeel Kazmi, Andrei Palade, Christian Cabrera, Gary White, and Siobhán Clarke. Dynamic service placement in multi-access edge computing: A systematic literature review. *IEEE Access*, 10:32639–32688, 2022.

Vincenzo Mancuso, Leonardo Badia, Paolo Castagno, Matteo Sereno, and Marco Ajmone Marsan. Efficiency of distributed selection of edge or cloud servers under latency constraints. In *21st Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 158–166, 2023.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Springer Science & Business Media, 1999.

Aritra Mitra, Hamed Hassani, and George Pappas. Exploiting heterogeneity in robust federated best-arm identification. *arXiv preprint arXiv:2109.05700*, 2021.

Aritra Mitra, Arman Adibi, George J Pappas, and Hamed Hassani. Collaborative linear bandits with adversarial agents: Near-optimal regret bounds. *Advances in Neural Information Processing Systems*, 35:22602–22616, 2022.

Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

Tao Ouyang, Rui Li, Xu Chen, Zhi Zhou, and Xin Tang. Adaptive user-managed service placement for mobile edge computing: An online learning approach. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1468–1476, 2019.

Athanasios Papoulis and S. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, Boston, 4th edition, 2002.

Haoran Peng and Li-Chun Wang. Energy harvesting reconfigurable intelligent surface for UAV based on robust deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 2023.

John G. Proakis and Masoud Salehi. *Digital Communications*. McGraw-Hill, 5th edition, 2008.

Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. Multi-objective multi-agent decision making: A utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–52, 2020.

Clémence Réda, Sattar Vakili, and Emilie Kaufmann. Near-optimal collaborative learning in bandits. *Advances in Neural Information Processing Systems*, 35:14183–14195, 2022.

Javad Sabzehali, Vijay K Shah, Harpreet S Dhillon, and Jeffrey H Reed. 3D placement and orientation of mmwave-based UAVs for guaranteed LoS coverage. *IEEE Wireless Communications Letters*, 10(8):1662–1666, 2021.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, 2020.

Shahin Shahrampour, Mohammad Noshad, and Vahid Tarokh. On sequential elimination algorithms for best-arm identification in multi-armed bandits. *IEEE Transactions on Signal Processing*, 65(16):4281–4292, 2017.

Megha Sharma, Abhinav Tomar, and Abhishek Hazra. Edge computing for industry 5.0: Fundamental, applications and research challenges. *IEEE Internet of Things Journal*, 2024.

Chengshuai Shi and Cong Shen. Federated multi-armed bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9603–9611, 2021.

Yao Shi, Emad Alsusa, and Mohammed W Baidas. On the application of uplink/downlink decoupled access in heterogeneous mobile edge computing. *Computer Networks*, 223: 109593, 2023.

Marta Soare. *Sequential resource allocation in linear stochastic bandits*. PhD thesis, Université Lille 1-Sciences et Technologies, 2015.

Marta Soare, Alessandro Lazaric, and Rémi Munos. Best-arm identification in linear bandits. *Advances in Neural Information Proceedings Systems*, 27, 2014.

Eckard Specht. The best known packings of equal circles in a square (up to n = 10000). http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html, 2022. (Accessed: 08.06.2025).

Jingcong Sun and Christos Masouros. Drone positioning for user coverage maximization. In *International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 318–322, 2018.

Péter Gábor Szabó, Mihály Csaba Markót, and Tibor Csendes. *Global Optimization in Geometry - Circle Packing into the Square*, pages 233–265. Springer US, Boston, MA, 2005.

Chao Tao, Qin Zhang, and Yuan Zhou. Collaborative learning with limited interaction: Tight bounds for distributed exploration in multi-armed bandits. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 126–146. IEEE, 2019.

Andrea Tirinzoni and Rémy Degenne. On elimination strategies for bandit fixed-confidence identification. *Advances in Neural Information Processing Systems*, 35: 18586–18598, 2022.

Harish Viswanathan and Preben E Mogensen. Communications in the 6G era. *IEEE Access*, 8:57063–57074, 2020.

Xiong Wang, Jiancheng Ye, and John CS Lui. Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach. In *IEEE International Conference on Computer Communications INFOCOM*, pages 1199–1208, 2022.

Xiong Wang, Jiancheng Ye, and John CS Lui. Online learning aided decentralized multi-user task offloading for mobile edge computing. *IEEE Transactions on Mobile Computing*, 23(4):3328–3342, 2023.

Yuanhao Wang, Jiachen Hu, Xiaoyu Chen, and Liwei Wang. Distributed bandit learning: Near-optimal regret with efficient communication. In *International Conference on Learning Representations (ICLR)*, 2020.

Vincent WS Wong, Robert Schober, Derrick Wing Kwan Ng, and Li-Chun Wang. *Key technologies for 5G wireless systems*. Cambridge university press, 2017.

Wenchao Xia, Tony QS Quek, Kun Guo, Wanli Wen, Howard H Yang, and Hongbo Zhu. Multi-armed bandit-based client scheduling for federated learning. *IEEE Transactions on Wireless Communications*, 19(11):7108–7123, 2020.

Wenchao Xia, Wanli Wen, Kai-Kit Wong, Tony QS Quek, Jun Zhang, and Hongbo Zhu. Federated-learning-based client scheduling for low-latency wireless communications. *IEEE Wireless Communications*, 28(2):32–38, 2021.

Yingce Xia, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Best action selection in a stochastic environment. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 758–766, 2016.

Liang Xiao, Xiaozhen Lu, Tangwei Xu, Xiaoyue Wan, Wen Ji, and Yanyong Zhang. Reinforcement learning-based mobile offloading for edge computing against jamming and interference. *IEEE Transactions on Communications*, 68(10):6114–6126, 2020.

Bo Xu, Wenchao Xia, Jun Zhang, Tony Q. S. Quek, and Hongbo Zhu. Online client scheduling for fast federated learning. *IEEE Wireless Communications Letters*, 10(7): 1434–1438, 2021.

Liyuan Xu, Junya Honda, and Masashi Sugiyama. A fully adaptive algorithm for pure exploration in linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 843–851. PMLR, 2018.

Mariam Yahya and Setareh Maghsudi. Joint coverage and resource allocation for federated learning in UAV-enabled networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2476–2481, 2022.

Mariam Yahya, Setareh Maghsudi, and Slawomir Stanczak. Federated learning in UAV-enhanced networks: Joint coverage and convergence time optimization. In *The 27th European Wireless Conference*, pages 1–7, 2022.

Mariam Yahya, Alexander Conzelmann, and Setareh Maghsudi. Decentralized task offloading and load-balancing for mobile edge computing in dense networks. *IEEE Communications Letters*, 28(8):1954–1958, 2024a.

Mariam Yahya, Setareh Maghsudi, and Slawomir Stanczak. Federated learning in UAV-enhanced networks: Joint coverage and convergence time optimization. *IEEE Transactions on Wireless Communications*, 23(6):6077–6092, 2024b.

Mariam Yahya, Aydin Sezgin, and Setareh Maghsudi. Service placement using distributed pure exploration in linear bandits. In *33rd European Signal Processing Conference (EUSIPCO)*, pages 2072–2076. IEEE, 2025.

Mariam Yahya, Aydin Sezgin, and Setareh Maghsudi. Service placement in small cell networks using distributed best arm identification in linear bandits. *IEEE Transactions on Mobile Computing*, pages 1–14, 2026.

Helin Yang, Jun Zhao, Zehui Xiong, Kwok-Yan Lam, Sumei Sun, and Liang Xiao. Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management. *IEEE Journal on Selected Areas in Communications*, 39(10):3144–3159, 2021a.

Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding. Federated learning via over-the-air computation. *IEEE Transactions on Wireless Communications*, 19(3):2022–2035, 2020.

Peng Yang, Ning Zhang, Shan Zhang, Li Yu, Junshan Zhang, and Xuemin Shen. Content popularity prediction towards location-aware mobile edge caching. *IEEE Transactions on Multimedia*, 21(4):915–929, 2018.

Peng Yang, Kun Guo, Xing Xi, Tony QS Quek, Xianbin Cao, and Chenxi Liu. Fresh, fair and energy-efficient content provision in a private and cache-enabled UAV network. *IEEE Journal of Selected Topics in Signal Processing*, 16(1):97–112, 2021b.

Peng Yang, Xing Xi, Kun Guo, Tony QS Quek, Jingxuan Chen, and Xianbin Cao. Proactive UAV network slicing for URLLC and mobile broadband service multiplexing. *IEEE Journal on Selected Areas in Communications*, 39(10):3225–3244, 2021c.

Shuangming Yang, Jiangtong Tan, Tao Lei, and Bernabe Linares-Barranco. Smart traffic navigation system for fault-tolerant edge computing of internet of vehicle in intelligent transportation gateway. *IEEE Transactions on Intelligent Transportation systems*, 24 (11):13011–13022, 2023.

Song Yang, Fan Li, Meng Shen, Xu Chen, Xiaoming Fu, and Yu Wang. Cloudlet placement and task allocation in mobile edge computing. *IEEE Internet of Things Journal*, 6(3):5853–5863, 2019a.

Zhaohui Yang, Cunhua Pan, Kezhi Wang, and Mohammad Shikh-Bahaei. Energy efficient resource allocation in UAV-enabled mobile edge computing networks. *IEEE Transactions on Wireless Communications*, 18(9):4576–4589, 2019b.

Changyan Yi, Jun Cai, Tong Zhang, Kun Zhu, Bing Chen, and Qiang Wu. Workload re-allocation for edge computing with server collaboration: A cooperative queueing game approach. *IEEE Transactions on Mobile Computing*, 22(5):3095–3111, 2021.

Tengchan Zeng, Omid Semiari, Mohammad Mozaffari, Mingzhe Chen, Walid Saad, and Mehdi Bennis. Federated learning in the sky: Joint power allocation and scheduling with UAV swarms. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

Chenxi Zhang, Pinyi Ren, and Qinghe Du. A contextual multi-armed bandit approach to caching in wireless small cell network. In *The 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2017.

Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, and Xinglin Zhang. Client selection for federated learning with non-IID data in mobile edge computing. *IEEE Access*, 9: 24462–24474, 2021.

Zhe Zhang, Shiyao Ma, Zhaohui Yang, Zehui Xiong, Jiawen Kang, Yi Wu, Kejia Zhang, and Dusit Niyato. Robust semisupervised federated learning for images automatic recognition in internet of drones. *IEEE Internet of Things Journal*, 10(7):5733–5746, 2022.

Ruiting Zhou, Xiaoyi Wu, Haisheng Tan, and Renli Zhang. Two time-scale joint service caching and task offloading for UAV-assisted mobile edge computing. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1189–1198, 2022.