ENGINEERING EDGE COMPUTING WITH PREDICTIVE INTELLIGENCE

SUPERVISORS:
DR. CHRISTOS ANAGNOSTOPOULOS
DR. SHAMEEM AHAMED PUTHIYA PARAMBATH

SALEH ALFAHAD

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING SCIENCE
COLLEGE OF SCIENCE AND ENGINEERING
UNIVERSITY OF GLASGOW

20th July 2026

Abstract

Distributed computing devices, including smartphones and edge micro-servers, gather and analyse data near users, presenting significant opportunities for enhancing service responsiveness and system efficiency. The constrained processing capabilities of edge nodes and the fluctuating nature of service requests pose considerable challenges in the management and orchestration of distributed services. This thesis investigates intelligent strategies for optimising decision-making in edge computing (EC) settings to improve Quality of Service (QoS), minimise latency, and ensure system flexibility under resource limitations.

The **first part** of the thesis introduces a time-efficient decision-making framework grounded in *Optimal Stopping Theory (OST)*, tackling the dilemma of whether edge nodes ought to replicate unavailable services (pull action) or delegate tasks to neighbouring nodes (push action). This system adjusts dynamically to variations in service request patterns and resource availability. The OST-based solution exhibits enhanced performance over conventional service management methods in synthetic experimental settings motivated by smart city applications.

The **second part** presents a cost-aware service orchestration model that enhances the OST method by including time-sensitive service demand patterns and system-level restrictions. This paradigm guarantees the provision of edge services in a resource-efficient and timely manner, reducing both over-replication and superfluous task offloading. It is assessed in several dynamic contexts to illustrate its practical efficacy in enhancing edge resource allocation.

The **third part** presents the initial study of *Experts with Gradient Bandits (ExpGradBand)* and addresses the important problem of node selection in Edge Computing settings. Through gradient-based decision-making and lightweight contextual experiments, it introduces *ExpGradBand*, a novel node-selection technique that extends the conventional *Multi-Armed Bandit (MAB)* framework. This part provides preliminary evidence of the algorithm's effectiveness through controlled, small-scale experiments with sparse contextual data. ExpGradBand outperforms baseline techniques in terms of convergence speed and cumulative regret, demonstrating adaptive decision-making under constrained experimental settings. These findings provide initial evidence of the framework's ability to support intelligent node selection in real time.

The **fourth part** builds on that initial research by providing a broader empirical validation of the same framework within a more comprehensive sequential learning setting that incorporates deep-learning expert models into the *Multi-Armed Bandit (MAB)* architecture. In uncertain and diverse edge contexts, this framework supports asynchronous, lost, and delayed feedback, enabling reliable and scalable decision-making. By adding expert-driven predictors, including Long Short-Term Memory (LSTM), Feedforward (FNN), and Recurrent (RNN) networks,

the *ExpGradBand* framework improves node-selection accuracy and strengthens adaptation to large-scale, real-world scenarios. This part therefore provides a broader experimental and analytical assessment of *ExpGradBand*, demonstrating its cross-scenario generalisation, scalability, and resilience in dynamic edge computing systems.

The **fifth part** extends the thesis by addressing dynamic assignment problems under strict budget constraints through a pure-exploration sequential learning framework. This chapter introduces a Thompson Sampling-based Recursive Block Elimination approach that efficiently explores large or continuous action spaces without requiring exhaustive sampling. By discretising the action space and recursively eliminating sub-optimal blocks using confidence-driven criteria, the proposed method enables reliable identification of near-optimal actions within limited budgets. The framework is evaluated on dynamic pricing and federated learning pruning scenarios, demonstrating strong performance in terms of misidentification probability and simple regret, and highlighting its scalability and effectiveness in resource-constrained decision-making environments.

The thesis closes by underscoring the need for adaptive, feedback-oriented orchestration systems in edge computing. This study advances the development of self-optimizing, resilient edge computing infrastructures by integrating Optimal Stopping Theory, cost-aware service modelling, and intelligent node selection algorithms, enabling the provision of efficient services within real-world operational limitations.

Contents

List of Tables

List of Figures

Acknowledgements

AlhamduliLlahi Robil 'Alamin!

First and foremost, all praise is due to **Allah**, whose guidance, strength, and mercy have enabled me to embark on and complete this journey.

I would also like to express my sincere appreciation to **the Saudi Prime Minister (Crown Prince Mohammad Bin Salman)**, **the Royal Saudi Air Force**, **Major General Abdulmonaim Alharbi**, **Senior Engineer Mohammad Aleissa**, **the Kingdom of Saudi Arabia**, and **the Saudi Arabian Cultural Bureau in the UK** for their generous support and encouragement throughout this journey.

I am deeply indebted to my supervisors, **Dr. Christos Anagnostopoulos** and **Dr. Shameem Ahamed Puthiya Parambath**, for their steadfast support, insightful guidance, and remarkable patience throughout the course of my studies. Their mentorship has played a pivotal role in shaping both my academic and personal development.

My heartfelt thanks go to **Dr. Roya Shadbakhsh** for her invaluable support during the early stages of my scholarship and travels. Her kindness and assistance made my transition smoother, and I remain truly grateful. I would also like to mention her young son, **Adrian**, a wonderful child, and I sincerely wish him a bright and successful future.

I would also like to acknowledge my brothers **Abdurhman**, **Hamad**, and **Ahmad**, whose unwavering encouragement, unconditional love, and constant belief in me have been a source of strength throughout this journey. Their endless support, thoughtful advice, and genuine care have inspired me to persevere and strive for excellence in every step of my academic and personal life. I am truly grateful for their presence and the motivation they have continuously provided.

I would also like to extend my heartfelt appreciation to my colleagues and friends, especially **Abdullah Aljaber**, my closest friend and confidant, who stood by me through challenging times with patience, understanding, and genuine friendship. His encouragement, humor, and companionship made even the most difficult moments lighter, and his support has meant more than words can express. I deeply value his friendship and the many memories we share along this journey.

Above all, I offer my deepest gratitude to my late father, my lifelong role model, whose sacrifices and dedication laid the foundation for everything I have achieved. His memory continues to inspire me every day. To my beloved mother, thank you for your boundless love, prayers, and unwavering support—your strength has been a constant source of comfort and motivation.

To my wife, **Anfal Albader**, and my children—**Munira**, **Lulu**, **Abdullah**, **Abdullmalik**, and **Roya**—thank you for your endless encouragement, patience, and love. Your belief in me has been the cornerstone of my perseverance, and I am forever grateful to share this journey with you.

Chapter 1

Introduction

1.1 Overview & Contributions

1.1.1 Motivation and Thesis Statement

The extensive use of end-user devices and the emergence of latency-sensitive, data-intensive applications—such as video analytics, smart city services, and intelligent surveillance—have necessitated computing paradigms that position processing nearer to data sources. **Edge Computing (EC)** has arisen as a remedy for the escalating issues associated with centralised processing, such as communication latency, network congestion, and data privacy problems [1]. By performing services and activities in proximity to data-generating devices, Edge Computing (EC) improves *Quality of Service (QoS)* by minimising latency, enhancing responsiveness, and facilitating localised decision-making.

Nonetheless, EC systems are intrinsically *resource-constrained*, functioning under constraints of computational power, storage capacity, and bandwidth. The limits are exacerbated by the dynamic and unexpected nature of service needs, since not all services can be pre-deployed on every node owing to capacity limitations [2]. In practical edge contexts, such as smart cities, service availability often falls short of demand, necessitating that nodes swiftly and judiciously choose how to manage incoming task requests in the absence of locally available services. The difficulty involves determining whether to reproduce the absent service at the local node (pull-action) or to outsource the duty to a neighbouring node or the cloud (push-action) [3].

This thesis presents a *time-optimized sequential decision mechanism* informed by **Optimal Stopping Theory (OST)** to resolve the decision-making conundrum. This methodology conceptualises service replication as a stopping problem, whereby an edge node monitors incoming requests and seeks to ascertain the optimal moment to intervene—either by duplicating the requisite service or by postponing action. The decision aims to optimise cumulative reward while reducing reaction time and replication expenses. This approach enables edge nodes to operate independently and adaptively in situations with variable service needs. The solution is

implemented via a one-stage look-ahead (1-sla) stopping rule, which assesses the current and prospective value of waiting compared to taking action [4]. Experimental assessments across several synthetic scenarios motivated by smart city applications indicate that the OST-based methodology substantially surpasses baseline solutions by attaining near-optimal service availability, minimising offloading delays, and enhancing resource efficiency. This challenge forms the basis for the first research question, **RQ1** (1.2.1), which investigates how edge nodes can autonomously make intelligent decisions between service replication and task offloading, based on demand patterns and time sensitivity. The proposed solution is described in **S1** (1.2.1) and explored in detail in **Chapter** 2.

In addition to personal choices about work delegation or service duplication, edge computing settings necessitate the *strategic orchestration* of resources across time, particularly in response to fluctuating demand circumstances [5]. A primary problem in this context is managing declining service demand rates attributed to factors such as user mobility, network variations, and fluctuating computational workloads. In Mobile Edge Computing (MEC) systems, the volume of tasks received at a certain node may diminish with time; yet, the timing of replication or offloading decisions significantly influences system efficiency and resource utilisation. This thesis presents a unique technique for cost-aware sequential decision-making, termed **DOST (Cost-based discounted sequential decision-making)** [6]. The methodology is founded on **Optimal Stopping Theory (OST)**, enhancing previous research by integrating the *temporal decay of task arrivals*, represented through exponential decline. DOST, in contrast to previous models that operate under a constant request rate, dynamically adjusts to diminishing task request patterns by assessing *when* to replicate services (pull-action) based on a *trade-off between anticipated reward and the cost of delay*. The decision-making process is mathematically articulated, representing task arrivals as a Poisson process with diminishing rates. The MEC node monitors the total number of service requests over time and uses a *1-stage look-ahead (1-sla)* strategy to ascertain the ideal stopping time t^* that maximizes the anticipated return. If no choice is made by the deadline T, the model defaults to task offloading (push-action), ensuring task continuance. Comprehensive simulations juxtapose the DOST methodology with the preceding COST model, demonstrating that DOST consistently attains superior payoffs, particularly in contexts characterised by variable or diminishing task demand. The findings validate that DOST surpasses baseline methods in both average performance and decision robustness, providing consistent payoffs and stopping behaviour despite fluctuations in capacity and demand degradation. This research forms the basis for the second research question, **RQ2** (1.2.2), which asks how EC nodes can make *cost-sensitive service orchestration decisions* under non-stationary demand. The proposed DOST-based solution is outlined in **S2** (1.2.2) and discussed in detail in **Chapter** 3.

Although node selection in Edge Computing (EC) has garnered significant attention, several existing solutions either depend on set heuristics or presume a static or stochastic environment, neglecting the dynamic and frequently hostile characteristics of actual edge deployments. In edge computing systems, the operational conditions of edge nodes such as availability, processing load, and communication quality can change rapidly, rendering effective, real-time task allocation a hard challenge. This complexity is increased by delayed, noisy, or absent feedback signals that undermine the reliability of conventional node selection algorithms. This thesis presents an enhanced feedback-driven node selection method that utilises gradient-based decision mechanisms and deep learning-based expert models within the **Multi-Armed Bandit (MAB)** framework to overcome these restrictions. The proposed method, designated ExpGradBand, augments the traditional Multi-Armed Bandit (MAB) approach by incorporating several autoregressive expert oracles, including Feedforward Neural Networks (FNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks, to deliver predictive insights regarding system state based on historical feedback signals, such as heartbeat and resource utilisation data [7]. In contrast to traditional methods, ExpGradBand employs a dual-layer selection process: it initially identifies the most promising expert based on historical predicted accuracy, followed by the implementation of a gradient bandit updating mechanism informed by the chosen expert's advice. This allows the system to adjust to both current performance feedback and changing contexts, thus optimising task allocation in real-time, even in hostile and non-stationary settings. Moreover, by utilising continuous-valued reward functions instead of binary outcomes, ExpGradBand provides enhanced precision in node evaluation and selection. A thorough experimental assessment utilising synthetic autoregressive system models reveals that ExpGradBand routinely surpasses leading methods such as Thompson Sampling, Exp3, Exp4, and fundamental Gradient Bandits. It attains the lowest cumulative regret, exhibits the steadiest convergence behaviour, and demonstrates resilient performance despite incomplete or noisy feedback. This contribution addresses the third research question, **RQ3** (1.2.3), which seeks to understand how *expert-driven, sequential learning mechanisms* can enhance node selection under uncertainty and dynamic system states. The proposed ExpGradBand-based solution is presented in **S3** (1.2.3) and elaborated in **Chapter** 4.

In edge computing (EC) systems, where computation is delegated from end devices to proximate nodes, a significant difficulty is the selection of suitable edge nodes for task execution. This difficulty stems from the inherent heterogeneity of edge nodes, which vary in computational capacity, availability, load, and network conditions. Conventional node selection techniques—relying on static heuristics or historical optimization—are inadequate for the requirements of dynamic, real-time systems. Specifically, these traditional methods lack the flexibility required to accommodate varying network circumstances, feedback delays, and changing task demands. This thesis examines sequential, feedback-driven learning processes for node selection in dynamic, non-stationary environments to address this gap. It presents **ExpGrad-**

Band, an innovative algorithm that incorporates *expert-driven contextual reasoning* into the **Multi-Armed Bandit (MAB)** framework [8]. In contrast to previous studies that presume stationary distributions or neglect feedback delays, ExpGradBand is specifically engineered for non-stochastic and adversarial contexts, characterised by unpredictable variations in node behaviour, availability, and load conditions. ExpGradBand utilises contextual feedback, including heartbeat signals and task performance metrics, and employs deep-learning time series predictors—such as Feedforward Neural Networks (FNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) models—as experts to inform the decision-making process. The system dynamically chooses both an expert and a node by integrating gradient-based learning and preference modelling, allowing for swift adaptation to feedback signals despite delayed or absent context. ExpGradBand has markedly reduced cumulative regret relative to previous bandit algorithms, as evidenced by comprehensive experimental assessments under diverse feedback settings, including delayed, absent, and noisy signals. It has notable resilience during initial operations and attains exceptional long-term adaptability, rendering it extremely appropriate for practical EC implementations where decision consistency and flexibility are crucial. This investigation forms the basis for the fourth research question, **RQ4** (1.2.4), which asks how edge nodes can be selected effectively under dynamic and uncertain conditions using real-time contextual feedback. The proposed solution is detailed in **S4** (1.2.4) and elaborated in **Chapter** 5.

In large-scale decision-making problems, particularly those arising in edge computing and distributed systems, identifying optimal actions under strict budget constraints poses a significant challenge. This difficulty is exacerbated when the action space is large or continuous and exhaustive exploration is infeasible. Traditional pure-exploration bandit algorithms typically require sampling all actions at least once, rendering them unsuitable for such constrained settings. This thesis addresses this limitation by investigating budget-aware, pure-exploration learning strategies for dynamic assignment problems. It introduces a **Thompson Sampling-based Recursive Block Elimination** framework that enables efficient identification of near-optimal actions without exhaustive exploration. The proposed approach combines Bayesian posterior sampling with confidence-driven block elimination to progressively discard sub-optimal regions of the action space while focusing exploration on promising areas. By discretising and clustering actions recursively, the framework achieves effective learning under limited budgets. Extensive experimental evaluations on dynamic pricing and federated learning pruning scenarios demonstrate that the proposed method significantly reduces misidentification probability and simple regret compared to state-of-the-art elimination-based bandit algorithms. This investigation underpins the fifth research contribution of the thesis and is detailed in **Chapter** 6.

1.2 Research Questions & Solution Overview

The thesis centres on creating intelligent, adaptive decision-making systems for effective service management and task execution in Edge Computing (EC) environments. It specifically tackles the difficulties of executing time-sensitive decisions for service replication and task offloading, cost-efficient orchestration amidst fluctuating demand patterns, and optimal node selection in uncertain conditions. This section delineates the principal research enquiries examined in the thesis and articulates the proposed resolutions for each.

1.2.1 RQ1: How can an edge node autonomously decide whether to replicate a missing service or offload a task when the requested service is unavailable locally?

S1: This method presents a sequential, payoff-optimized technique grounded on **Optimal Stopping Theory (OST)** to inform the choice between duplicating a service (pull) and offloading a task (push). The technique simulates request observations across time and employs a 1-stage look-ahead (1-sla) strategy to optimise system rewards and minimise replication delays, resulting in enhanced and autonomous service management at the edge (detailed in **Chapter** 2).

1.2.2 RQ2: How can EC nodes make cost-aware service orchestration decisions under conditions of declining or time-varying demand?

S2: This system offers a dynamic service management strategy based on the **DOST** model. It integrates time-decaying service request rates to ascertain the most cost-effective timing for service replication prior to transitioning to task offloading. The method adjusts to variations in request frequency and decision urgency, enhancing resource utilization and long-term benefits (detailed in **Chapter** 3).

1.2.3 RQ3: How can expert-based bandit models further improve task placement accuracy in real-time EC systems with noisy, incomplete feedback?

S3: This method presents an initial, reduced-scale implementation of the ExpGradBand framework, evaluated through preliminary contextual-bandit experiments. It enhances the basic Gradient Bandit algorithm by integrating lightweight autoregressive expert predictors (FNNs, RNNs, LSTMs) into the MAB selection process. It employs gradient bandit optimisation to integrate

expert knowledge with real-time adjustment, enabling robust yet small-scale node selection validation under controlled conditions. The technique exhibits faster convergence and lower cumulative regret in a simplified experimental setup, offering preliminary evidence of the framework's potential (detailed in **Chapter** 4).

1.2.4 RQ4: How can edge systems select the most suitable node for executing a task under dynamic and uncertain feedback conditions?

S4: This solution extends the preliminary ExpGradBand study into a comprehensive, large-scale investigation. It builds on the **ExpGradBand** framework introduced in Chapter 4. This gradient-based multi-armed bandit (MAB) method is augmented with delayed, lost, and asynchronous contextual feedback as well as deeper expert architectures. The approach works in adversarial and non-stationary EC contexts, using past signals and acquired patterns to adaptively choose optimal nodes across heterogeneous and dynamic edge environments, even when feedback is noisy or incomplete. Compared with Chapter 4, this extended study introduces broader datasets, longer horizons, and advanced analyses, validating ExpGradBand's robustness and scalability (detailed in **Chapter** 5).

1.2.5 RQ5: How can dynamic assignment problems be solved efficiently under strict budget constraints in pure-exploration settings?

S5: This solution investigates budget-aware pure-exploration learning strategies for large-scale and continuous action spaces, where exhaustive exploration is infeasible. It introduces a **Thompson Sampling-based Recursive Block Elimination** framework that combines Bayesian posterior sampling with confidence-driven elimination to progressively discard sub-optimal regions of the action space. The proposed approach enables efficient identification of near-optimal actions under limited budgets by recursively clustering and eliminating action blocks. Extensive experimental evaluations on dynamic pricing and federated learning pruning scenarios demonstrate significant improvements in misidentification probability and simple regret compared to state-of-the-art elimination-based bandit algorithms. The full methodology and analysis are presented in **Chapter** 6.

1.2.6 Academic Contributions to the Research Community

The work presented in this thesis is primarily derived from published research conducted by the author in collaboration with other researchers. The chapters and the corresponding papers on which these chapters are based are listed below.

1. Chapter 2: Time-Optimized Sequential Decision Making for Service Management in Smart City Environments

- **Saleh ALFahad**, Christos Anagnostopoulos, Kostas Kolomvatsos, "Time-Optimized Sequential Decision Making for Service Management in Smart City Environments." *Journal of Smart Cities and Society*, vol. 1, no. 4, 2022, pp. 277–299.

2. Chapter 3: Task Offloading in Mobile Edge Computing Using Cost-Based Discounted Optimal Stopping

   - **Saleh ALFahad**, Qiyuan Wang, Christos Anagnostopoulos, Kostas Kolomvatsos, "Task Offloading in Mobile Edge Computing Using Cost-Based Discounted Optimal Stopping." *Open Computer Science*, vol. 14, no. 1, 2024, article 20230115.

3. Chapter 4: Gradient Bandit Experts for Node Selection in Edge Computing

   - **Saleh ALFahad**, Shameem Puthiya Parambath, Christos Anagnostopoulos, "Gradient Bandit Experts for Node Selection in Edge Computing." *Proceedings of the 45th IEEE International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2025.
   - **This paper received the Best Poster Paper Award at the 45th IEEE International Conference on Distributed Computing Systems (ICDCS 2025), held in Glasgow, United Kingdom.**

4. Chapter 5: Node Selection Using Adversarial Expert-Based Multi-Armed Bandits in Distributed Computing

   - **Saleh ALFahad**, Shameem Puthiya Parambath, Christos Anagnostopoulos, Kostas Kolomvatsos, "Node Selection Using Adversarial Expert-Based Multi-Armed Bandits in Distributed Computing." *Computing*, vol. 107, article 85, 2025.

5. Chapter 6: Thompson Sampling-based Recursive Block Elimination for Dynamic Assignment under Limited Budget in Pure-Exploration

   - Shameem Puthiya Parambath, Christos Anagnostopoulos, **Saleh ALFahad**, "Thompson Sampling-Based Recursive Block Elimination for Dynamic Assignment under Limited Budget in Pure-Exploration." *Data Mining and Knowledge Discovery*, vol. 39, no. 1, article 7, 2025.

1.3 Thesis Structure

The thesis is organised into seven major chapters. The Introduction outlines the research motivation, main contributions, and research questions that guide the thesis. Chapters 2, 3, 4, 5, and 6 form the main body of the thesis. Chapter 2 delineates a sequential decision-making framework based on optimum stopping theory for the replication of intelligent services and the offloading

of tasks. Chapter 3 presents a cost-efficient service orchestration methodology using a discounted optimum stopping model to address fluctuating demand. Chapter 4 expands this concept by including a gradient-based bandit strategy augmented with deep expert predictors to provide resilient task allocation in the presence of noise. Chapter 5 addresses node selection in uncertain and dynamic environments, using adversarial multi-armed bandits with expert input. Chapter 6 introduces a Thompson Sampling-based Recursive Block Elimination framework for dynamic assignment problems under limited budget constraints in pure-exploration settings. The chapter proposes a confidence-driven block elimination strategy and evaluates its effectiveness on dynamic pricing and federated learning pruning scenarios. Chapter 7 culminates the thesis by summarising major results, delineating present limits, and exploring future research avenues in intelligent edge service management.

1.4 Thesis Roadmap

To make the thesis logic immediately visible, Table 1.1 provides a compact roadmap that links each research question (RQ1–RQ5) to its corresponding solution (S1–S5), the chapter where it is developed, and the main type of evaluation evidence used to validate it. This mapping clarifies how the thesis contributions progress from problem formulation to algorithmic design and empirical/theoretical validation, and it enables the reader to directly trace each claim to the strongest supporting result.

The roadmap is aligned with the “Research Questions & Solution Overview” section, and it highlights where each contribution is introduced and how it is evidenced in the thesis.

As shown in Table 1.1, Chapters 2-6 each correspond to one core research question and provide a dedicated solution with supporting evidence, while Chapter 7 consolidates the results and outlines limitations and future directions.

Item	Description
RQ1	Service replication vs. task offloading using Optimal Stopping Theory (OST).
S1	OST-based sequential decision model (pull vs. push action).
Chapter 2	Time-optimized service management with theoretical OST framing and simulation-based evaluation.
RQ2	Cost-aware service orchestration under time-varying and declining demand.
S2	Discounted Optimal Stopping Theory (DOST).
Chapter 3	Discounted OST model validated via comparative simulations and sensitivity analysis.
RQ3	Expert-assisted node selection under noisy and limited feedback (proof-of-concept).
S3	Reduced-scale ExpGradBand with lightweight expert predictors.
Chapter 4	Preliminary contextual-bandit experiments demonstrating convergence and regret gains.
RQ4	Robust node selection in adversarial and non-stationary environments.
S4	Full-scale ExpGradBand with delayed and lost feedback handling.
Chapter 5	Large-scale experimental validation under adversarial feedback conditions.
RQ5	Efficient pure-exploration under strict budget constraints.
S5	Thompson Sampling-based Recursive Block Elimination.
Chapter 6	Experimental evaluation on dynamic pricing and federated learning pruning scenarios.

Table 1.1: Thesis roadmap mapping research questions to solutions, chapters, and validation evidence

Chapter 2

Time-optimized Sequential Decision Making for Service Management in Smart City Environments

2.1 Introduction

As a result of the fast growth of the Internet of Things (IoT), smart cities are replete with linked sensors and computing devices used for information collection and processing [9]. Various collaborative services (e.g., smart medical assessment, smart university, elderly support) emerge as a result of the right processing and utilization of the huge data created by the Internet of Things [10]. On the basis of integrated services, it is likely that the smart city will be able to increase the Quality of Experience (QoE) of people while reducing the use of resources. Furthermore, smart cities are endowed with a robust capacity to observe, analyze, and adapt to their residents [11]. Nevertheless, because of the rapid growth of data and the limited processing capacity of endpoints, it is challenging to develop joint services that are typically time and latency-sensitive when depending purely on IoT devices. To increase the QoE in smart cities, it is essential to use additional computational resources [12]. As a result, Edge Computing (EC), which leverages computational resources at the network edge and can meet real-time demands, has been proposed as an effective paradigm to resolve the issue of inadequate computing resources in smart cities [13]. Figure 2.1 illustrates a typical edge computing environment, including IoT devices and users, local edge nodes, neighbouring edge nodes, and cloud data centers.

The expansion of IoT and edge computing has enabled a large number of devices and processing nodes to be located close to end users, supporting ubiquitous and pervasive services [14]. In recent years, we have also seen the introduction of several mobile devices and applications, such as drones and Vehicular Networks (VN), each with its own unique set of capabilities. The development of this technology has made it possible for computing and sensing devices to launch applications such as Augmented Reality (AR), predictive analytics activities at the edge,

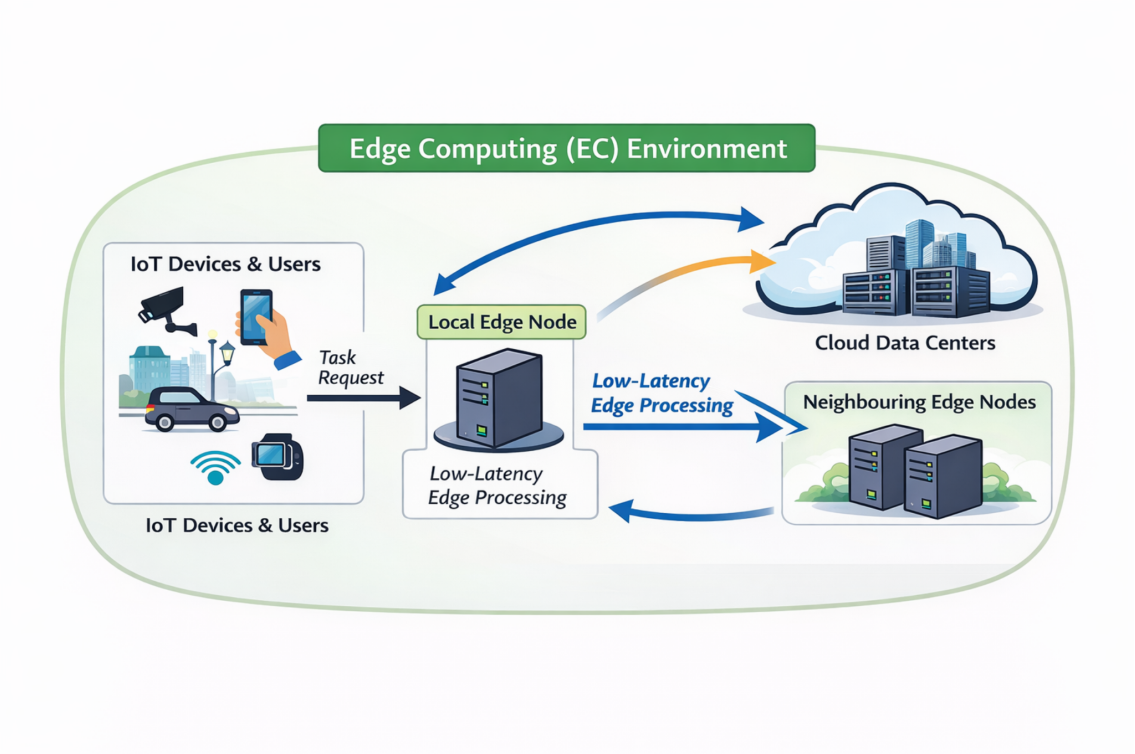


Figure 2.1: Illustration of an edge computing environment with IoT devices, edge nodes, and cloud data centers.

intelligent vehicle control, traffic management, and interactive applications [15]. These types of services are given in response to the need for processing data being gathered by IoT devices and, then, being sent to edge nodes [16]. A range of IoT applications may provide end users with more precise and detailed network services [17]. In this situation, an increasing number of equipment and sensors are being networked through IoT technology, and this equipment and these sensors will create vast amounts of data that need additional processing, therefore delivering service providers and end users with intelligence. In the traditional Cloud computing setup, all information must be transferred to centralized servers, and after processing, the outcomes must be sent directly to the requestors, e.g., sensors or other devices. This procedure places significant stress on the network, particularly in terms of bandwidth and resource requirements for data transmission [18]. Furthermore, as data volumes increase, bandwidth demand increases, and network congestion intensifies.

This paradigm has been suggested to address the aforementioned difficulties [19], [20]. The fundamental goal is to increase the processing capabilities at the network edge; specifically, computing [21], bandwidth [22], and storage resources [23] are relocated nearer to IoT systems in order to minimize core data traffic and response delay and to support resource-intensive IoT applications. In comparison to Cloud [24] [25], edge nodes have limited storage and computing capabilities. Nonetheless, they are able to host a number of services that make it easier to carry out a variety of processing tasks [26]. These kinds of activities are often presented in the

form of assignments that 'request' the processing of data (e.g., predictive analytics, explanatory-driven models). The integration of IoT and edge computing to facilitate the implementation of collaborative activities in which nodes can collaboratively complete the tasks that have been requested is an intriguing development [27]. For instance, nodes share services or data and they even offload processing duties onto peers in certain circumstances. Because of their reduced capability for storage and computing, edge nodes are only able to host a (sub-)set of the total accessible services and the data that has been obtained. The collected data can be reported by stream monitoring e.g., the evolution of a certain phenomenon [28], [29], [30] and streams generated by autonomous nodes (e.g., unmanned vehicles) [31]. Upon these streams, we can support the extraction of knowledge, the detection of events, or any other processing according to the application requirements [14].

Services are essential in order to perform the processing responsibilities that have been delegated by applications or users in the form of tasks. As a result, the demand for services is always adjusting to accommodate the dynamic requests of various activities/tasks. This indicates that a task can be asking for a service that may or may not be available locally. Then, nodes have two options:

- Option 1: They can perform a service replication (*pull-action*) from their edge neighbors or the Cloud, but only if the service is suitable for their computing capabilities;
- Option 2: They can delegate the responsibility (*push-action*) to the peers/Cloud that presently host the service(s).

The aforementioned options deal with deciding where to transfer service in order to keep the Quality of Service (QoS) at a high level. Because of the nature of the edge environment and the non-static behavior of end users, it is difficult to provide an ideal migration/replication plan that can be implemented. Moreover, due to the situations that make the local authority of the user complex, offloading tasks involves sending them to peers or the Cloud for processing. For instance, the node that is assigned the duty of processing as the receptor of a task could have a high load, and as a result, it might not be able to guarantee an effective completion within the permitted time interval that may be specified by the requester. A strategy for the migration of services was presented in [32]; its purpose is to satisfy the service latency requirements of EC while keeping migration costs and trip times to a minimum. Furthermore, in [33], the authors recommended using a reinforcement learning-based model to solve the service migration issue.

In this chapter, we focus on the pull-action leaving the initiative to edge nodes to enhance their autonomous nature, and suggest that services' replication may be optimized and controlled in the edge environment by adopting the principles of Optimal Stopping Theory (OST) [34]. In this case, these nodes must decide locally when it is the *best* time to replicate the service in order to maximize its ability to carry out the requested tasks. The service-replication decision is made independently by each edge node, in accordance with the requirements of the incoming

tasks and the current state of the node. This chapter addresses the first research question of the thesis (RQ1), namely how an edge node can autonomously decide whether to replicate a missing service or offload a task when the requested service is unavailable locally. The rest of this chapter is organized as follows. In Section 2.2, we provide a summary of the prior work as well as the rationale and our contribution, while Section 2.3 delves into the specifics of the proposed OST-based decision-making model. The outcomes of our performance and comparative assessment are presented in Section 2.4, and Section 2.5 draws the conclusion of this work and indicates potential avenues for further research.

2.2 Related Work & Contribution

2.2.1 Related Work

The majority of approaches for task offloading at the edge depend on the selection of whether or not the task should be processed locally or should be offloaded to peers or the cloud. The time in execution latency [35] and the amount of energy used should both be reduced as much as possible. These are the two primary goals. The work presented in [36] provides a framework for offloading computation workloads from a user device to a server hosted in EC. Our approach is different from that due to the fact that our time-optimized decision primarily considers the actual benefit of payoff which indicates the current users' requests for specific service. In [36], the User Equipment (UE) choice on whether to offload computing tasks with the highest CPU availability for a certain application is driven by the radio network information service (RNIS), according to the current value of round trip time (RTT) between the UE and the edge node. In our method, the offloading choice ideally relies on the number of users' requests for specific service as well as the cost of delay. Therefore, the decision of offloading is made when the dedicated time of the required service by users is finished which indicates the required service is not worth being hosted. As a result, this avoids unnecessary service hosting and improves resource utilization at the edge.

The work in [37] presents a method for optimizing the decision-making process that an Unmanned Aerial Vehicle (UAV) uses to choose whether or not to do the compute task locally or to offload it to an edge server. The decision is made based on a series of interactions that take place between the UAV and IoT systems. During these interactions, the UAV receives feedback on the state of the network and EC server, which enables an estimation of the remaining amount of time needed to complete the task. As a result, the UAV uses this information to solve an optimization problem with the purpose of decreasing a weighted sum of delay and energy costs as much as possible. In our work, EC chooses the ideal moment that has a high payoff to perform a service replication (pull-action) from their edge neighbors or the cloud. The approach described in [38] analyses the present state of the node's resources, which are modeled as a

Markov Decision Process utilizing Q-learning for training, in order to identify which portion of the application ought to be offloaded and then moves on with an offloading decision. The primary objective is to reduce the delay experienced by offloaded applications while taking into account whether computation should be offloaded to a nearby fog node, i.e., an intermediate computing resource located between end devices and the cloud, or to the cloud. In our work, we examine a scenario in which the edge node only performs service replication (pull-action) from neighbouring edge nodes or the cloud, since the node is static and makes this decision with minimal delay under realistic operating conditions.

The work in [39] focuses on reducing processing delay and energy consumption at the edge device by optimizing task allocation and the central processing unit (CPU) frequency. In contrast, our study focuses on decision-making based on the popularity of the requested service in order to determine whether service replication or task offloading should be performed. The authors of [40, 41] investigate task-offloading decisions in multi-user and multi-edge environments. By contrast, our approach considers a single-user setting and focuses on determining the pull-or-push action for an edge node or a peer, which is more suitable for low-capacity settings. The study in [42] considers task execution under a time-division multiple access protocol, where edge users may execute tasks locally or offload all or part of them to the access point (AP), with the objective of reducing the AP's overall energy consumption. In contrast, our proposed solution is guided by service popularity and an OST-based stopping rule that determines the action with the maximum payoff. Finally, [43] addresses task offloading in ultra-dense networks with the aim of reducing latency while preserving the battery life of user equipment. Unlike this line of work, our method determines the optimal pull-or-push decision for service migration between neighbouring edge nodes or peers.

The work in [44] and [45] primarily formulate the decision-making of offloading based on a resource scheduling factor. However, in our approach, we develop the decision of (push or pull - action) w.r.t accumulation of the newly received tasks and cost incurring. A state-action-payoff-state-action (RL-SARSA) method is introduced in [46] to tackle the resource management issue at the edge server and make the best offloading option for reducing system cost. Moreover, the work in [47] aims to make the offload decision based on minimizing the cost. However, our work emphasizes making the decision with the greatest payoff, considering the expense cost each time that we do not make the decision of (push-action). In the study presented in [48], [49], the issue of offloading computation for many users in a wireless environment with uncertainty is investigated. Nevertheless, the goal of our work is to provide the single-edge user with the best possible time in which to make a decision (push or pull - action). In [50] [51], the authors examine the challenge of partial offloading scheduling and resource allocation for edge computing systems with various independent activities. The objective is to reduce the weighted total of processing latency and energy usage while satisfying the tasks' transmission power limitations. Nevertheless, our approach enables the edge user to choose the optimal

(pull-action) solution. Since the service has previously been hosted, the wait for future tasks will be reduced. The work in [52] and [53] investigate energy-efficient task offloading in edge computing. The goal is to reduce the amount of energy required for task offloading. However, our method focuses on making the choice at the optimal moment to host the most popular tasks, hence optimizing the edge's capacity resource utilization.

Migration of data and services may be used to fill 'gaps' in the local knowledge base of a processing node. In [54], the interested reader may obtain a survey of related initiatives. When migrating services, it will be efficient to meet processing demands locally; nonetheless, a list of factors must be considered. To minimize overburdening the network, service migration may be accomplished in phases; service components should be "hooked up" to the hosting infrastructure swiftly, easily, and automatically. One of the technologies used for these reasons is Machine Learning (ML). ML may serve as the foundation for producing models that predict the movement of users and provide a proactive procedure for services migration [55]. Reinforcement learning may improve agent behavior while determining the optimal action to take [56], [57]. The difficulty in depending on supervised machine learning technology is identifying the appropriate representative training data set. As a result of the raised degree of uncertainty in the corresponding decision-making, it is challenging to gather data that includes all alternative node statuses. The service migration model may alternatively be expressed as an online queue stability issue [58]. The issue may be addressed using either a Lyapunov optimization or a multi-objective optimization scheme [59], [60], and the Pareto optimum solution can be determined. In the past, efforts have mostly focused on minimizing delay within the restrictions of energy usage in EC settings. However, the bulk of research efforts that aim at a user-centric service migration model is 'affected' by the extra time required to address the optimization issue, except if a list of assumptions is made or approximate solutions can be accepted. Using Markov chains is also a possible solution to settle the issue. Using a modified policy-iteration algorithm, the authors of [61] provide a solution to a finite-state Markov decision process. A technique based on Thompson sampling is suggested in [62] to facilitate dynamic service migration choices. The authors of [63] describe a service allocation module for networked automobiles powered by Cloud-based services. The objective is to increase the amount of autonomy of automobiles capable of picking nearby automobiles to share adopted services. The authors of [35] suggest a proactive system for determining the effective administration of services and tasks that are present/reported at EC nodes. The suggested model monitors the demand for existing services and rationalizes their management, i.e., their local presence/invocation when the demand is modified by the desired processing operations. To proactively achieve the strategic objectives of the envisioned model, a statistical inference process is initiated for each incoming task.

2.2.2 Rationale & Contribution

Figure 2.2 illustrates in a simplified way the problem of deciding whether to perform a service replication (pull-action) from their edge neighbors or the cloud or offload (push-action) it to the cloud service to make the necessary computation to complete the required task. Let us assume that the starting time of recording the number of requests for a task is $t = 0$ and the end time period, hereinafter referred to as deadline is $T > 0$, where a decision should be made. Figure 2.2 shows two decisions against time. Specifically, in Figure 2.2 (a), we observe the scenario of making the pull-action decision at an early time t^* (and before the deadline T) since we are expecting receiving more tasks requesting a specific service. This denotes that there is a relatively high demand for this service, thus, it is advisable to decide on pulling the service locally to the node. In Figure 2.2 (b), we observe the scenario in which we delayed the decision making in light of increasing our confidence that there is no high demand of the service. Once the deadline elapses and we have not decided on a pull-action, which indicates that the service request rate was relatively low, then, the node decides on offloading the service requests (push-action) either to its peers or the Cloud. In other words, this implies that there was not a large volume of tasks requesting the specific service, thus, offloading these tasks is necessary. Assume the case where we made the decision to download/pull the service earlier enough, and then after a while, we realized that the service was requested only by a small number of users. We would have consumed additional cost of resources without benefiting from the download/pull service. As a result, it would have been more efficient to delay our decision until the end of the dedicated time and then perform the action of offloading as it is shown in Figure 2.2 (b). On the contrary, if we had decided to send/offload the request to the Cloud at an early time, then, we would not have high confidence whether this request was highly requested on the node, thus, this decision would not be efficient. The reason is that this service would have been highly requested in the future, thus, the push-action would not be appropriate. However, if after a while we realize that the service demand has significantly increased, then an early push-action results in overloading the network with offloading tasks to other peers or the Cloud unnecessarily. Thus, an appropriate process is to delay the decision in a principled manner and monitor the history of task requests over time before performing service replication (pull-action) from neighbouring nodes or the cloud to the local node at the optimal time t^*, as illustrated in Figure 2.2(a). The dotted horizontal line in the figure denotes an illustrative popularity threshold used to visually distinguish high-demand and low-demand cases, while the decision itself is determined by the observed request history and the stopping criterion evaluated over time. Hence, the problem is to identify the optimal decision time t^* so that resources are utilized efficiently while balancing the trade-off between task offloading and service replication. A fundamental solution is therefore proposed to ensure that the appropriate decision is taken at time t^* while accounting for delay costs.

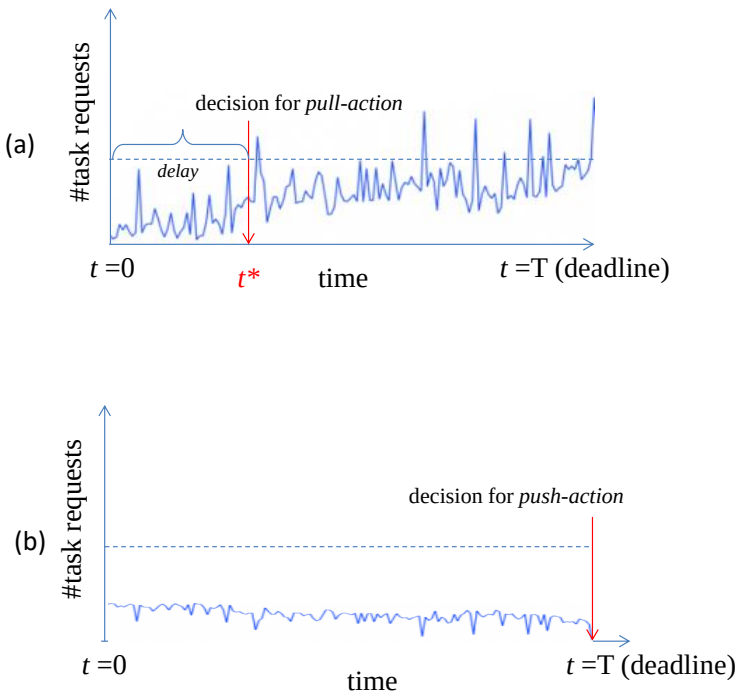


Figure 2.2: Illustration of the observed task request rate over time for distinguishing between (a) high-demand and (b) low-demand service popularity cases. The dotted horizontal line denotes an illustrative popularity threshold, while the decision is determined by the observed request history and the stopping criterion evaluated over time.

Contribution: Our approach is adaptable to applications that use offloading decision-making algorithms in EC environments. Our major technical contribution is:

- We provide a time-optimized intelligent method that enhances the likelihood of making the decision of service replication (pull-action) of the service with the greatest payoff. This is reflected by the cumulative sum of the number of task requests to the node under consideration taking into account the incurred cost.
- We provide theoretical analysis of the optimality of our model based on the principles of the Optimal Stopping Theory.
- We report on a detailed comparative evaluation of our OST-based method against the Ideal Decision ruLe (IDL), which produces the actual maximum payoff, and a heuristic Deterministic Rule (DR).
- A comparative assessment is provided comparing our results with the related work in [64], which is based on the Secretary Optimal Stopping Time problem.

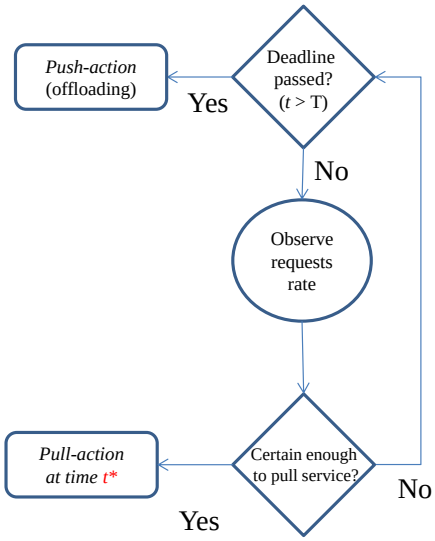


Figure 2.3: Sequential decision-making diagram on either task offloading (push-action) or service replication (pull-action) at the best time $t^* < T$ by the edge node.

In addition, Figure 2.3 illustrates the decision process of the proposed mechanism. Specifically, it shows how the node monitors incoming requests over time in order to identify the optimal decision time for either service replication (pull-action) or task offloading (push-action). Before either terminal action is taken, the node remains in its current steady state, continues handling incoming requests, and re-evaluates the decision at the next time instance. The decision is therefore influenced by the observed request history for a specific service at each time instance t.

The rationale is as follows. The node receives a number of incoming task requests Y_t at every time instance $t < T$. Then, the node has two options:

- If the number of received requests is high enough, the decision will be to pull the service at a time $t \leq t^* < T$.
- Otherwise, the node reconsiders its decision at the next time $t + 1$, where new service requests are coming.

If the deadline T elapses, then the decision will be to push the task to a peer or the Cloud for further processing. To the best of our knowledge, this is the first study to formulate sequential pull/push decision-making as an OST problem in edge computing environments. The overall goal is to increase the possibility of making the decision of service replication (pull-action) with the greatest payoff. Moreover, since the nodes are located among the servers that have been distributed at the edge of the network, our approach will be applied to EC applications such as Virtual Networks (VN) [65], UAV [15], and data mining applications [14].

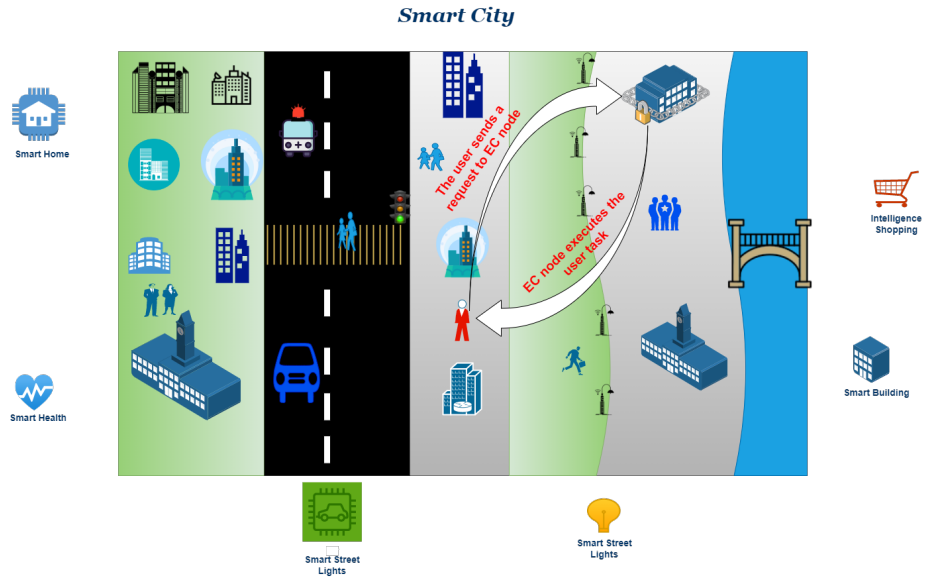


Figure 2.4: Example of the two actions: pull and push in a Smart City environment.

2.3 System Model & Problem Formulation

In Section 2.3.1, we begin by providing an overview of the system model, and then in Section 2.3.2 we discuss the problem definition. In Section 2.3.3, the Optimal stopping rule problem is introduced. Finally, in Section 2.3.4, we will give full details about cost-based sequential decision-making.

2.3.1 System Model

As seen in Figure 2.4, a smart city is increasingly establishing vast IoT networks with widely distributed IoT devices that provide vast amounts of services. In light of the vast size and extensively spread nature of IoT networks, EC has become a strong and efficient paradigm for providing processing capabilities to IoT devices at the network's edge [66]. In EC, IoT services may be executed on EC, which offers low delay and reduces the bandwidth load. Nevertheless, it remains difficult to enhance the entire EC execution performance, for instance, resource utilization, EC energy usage, and load balance. However, we believe that the environment of the

network is made up of a global cloud data center and an EC system [67] [68]. The EC system is composed of N base stations, as defined in Table 2.1, where each base station is associated with one or more EC servers. In this model, the base stations provide connectivity to nearby edge devices, whereas the associated EC servers provide the computational and storage resources required for service execution. Together, a base station and its associated EC servers form an EC node. Accordingly, we will refer to the set of EC nodes as $\mathcal{N} = [1,2,...,N]$. Our general assumption is that multiple service providers would deploy their own applications on each EC node. Each edge device has the ability to delegate tasks to an EC node that is equipped with appropriate computing services. In addition, edge devices have the capability of offloading tasks to a global cloud, presuming that the global cloud has all of the required services and a decent amount of computing resources [69]. Edge devices make the decision of task service replication (pull-action) in accordance with the maximum payoff value. The time is segmented into frames denoted by the notation $t = 1,2,\ldots,T$. A decision of service replication (pull-action) is determined by the edge device at some moment in time t^*, depending on local information (e.g., task information like the number of services that have been received). We rely on the assumption that the location of the edge device and the network environment remain the same throughout each time instance. This allows us to guarantee that the task of replication (pull-action) decision will be effective at the same time instance t.

2.3.2 Problem Definition

In the task model, we assume that task requests arrive according to a Poisson process with rate λ. Let Y_t denote the number of task requests observed at time instance t [70]. Since the edge device is assumed to remain stationary, the set of EC nodes within its sensing and communication range remains unchanged during each time instance t. Accordingly, the cumulative number of task requests up to time t is computed over the requests received by the edge device. The service replication (pull-action) decision is determined by the threshold θ, which represents the number of task requests used by the OST-based model to trigger the decision. The cumulative distribution function of the Poisson distribution can be expressed as in (2.1).

$$P(Y <= y;\lambda) \quad = \quad \sum_{k=0}^{y} \frac{e^{-\lambda} \cdot \lambda^k}{k!} \tag{2.1}$$

We define the cumulative number of requests R_t, for $0 \leq t \leq T$, as:

$$R_t = \sum_{k=0}^{t} Y_k. \tag{2.2}$$

Here, the Poisson model gives the probability of observing y task requests at a given time instance under arrival rate λ. Based on these observations, the cumulative number of requests R_t is monitored over time, and the decision is taken so as to stop as close as possible to the threshold θ without exceeding it; otherwise, task offloading (push-action) is triggered when waiting longer is no longer beneficial.

The objective is to stop at a time when the cumulative number of observed requests is as close as possible to the threshold θ without exceeding it. If $R_t > \theta$, the observation is immediately terminated and the service offloading scenario is activated, since the cumulative number of requests has exceeded the target threshold. The second scenario is not itself undesirable; however, exceeding the threshold represents an outcome that the decision mechanism aims to avoid. Therefore, when $R_t > \theta$, a penalty is applied by assigning zero payoff, which encourages the decision to be made before the cumulative number of requests exceeds θ. However, we may include a penalty for the increased latency caused by receiving one more task before stopping. Each time we do not stop, hence not triggering the service replication (pull-action) scenario, we incur a cost greater than zero. This cumulative cost up to t requires the procedure to decide whether to stop or continue with another observation by evaluating the trade-off between extending the observation for potentially more usable data and the extra latency required to activate the service offloading scenario. This price may reflect the urgency of a monitoring application that demands a service offloading option in near real-time. Alternatively, if the application requires very precise service offloading option outcomes, a (limited) delay must be permitted.

Based on the above interpretation, we define the following payoff function $H_t(R_t)$, which involves (i) the task threshold θ, and (ii) the delay cost per monitoring step $C \geq 0$, where C is a constant cost parameter.

$$H_t(R_t) = \begin{cases} R_t - C \cdot t & \text{if } R_t \leq \theta \\ 0 & \text{if } R_t > \theta \end{cases} \tag{2.3}$$

Note that the payoff function may become negative when the accumulated delay cost exceeds the observed number of requests (i.e., when $C \cdot t > R_t$). This reflects that continuing the observation is no longer beneficial. The payoff is evaluated only up to the stopping time. Once the cumulative number of requests exceeds the threshold (i.e., $R_t > \theta$), the observation process is terminated, and no further payoff is accumulated.

The rationale behind this payoff is that we desire the rate of service demands per time t, taking out the delay cost to be significantly increasing. This expresses the likelihood that the pull-action is stochastically a better decision than the push-action. This is because we increase our confidence that in the (near) future, we are expecting more requests for a service in a node.

Therefore, we aim at maximizing this payoff and, hence, the corresponding likelihood. Our goal is to attain θ by maximizing the number of received tasks. If R_t is greater than θ, then the return is zero since the mechanism should have triggered the service offloading scenario. Therefore, any additional delay was of no benefit.

It is mandatory to have a base result to be compared with our fundamental approach (OST). Therefore, we define the Ideal Decision ruLe (IDL), which measures the cumulative sum of the received tasks from users to the node. However, each time instance t, the expected cost $C \cdot t$ must be taken into account. As a result, the instance time t that has the maximum payoff that results from IDL is the best time t^* instance for making the decision (pull or continue). Moreover, we can express the IDL best time instance t^* as (2.4):

$$t^* = \{0 \leq t < T : \max_t \sum_{k=0}^{t} Y_k - C \cdot k, \text{ s.t. } \sum_{k=0}^{t} Y_k \leq \theta\} \tag{2.4}$$

Here, Y_k denotes the number of task requests observed at time instance k, which follows the Poisson arrival process defined earlier. The objective is to determine the time instance t^* that maximizes the cumulative reward $\sum_{k=0}^{t} Y_k - C \cdot k$, while ensuring that the cumulative number of requests remains below the stopping threshold θ. In general, these decision rules determine the appropriate time to trigger the service offloading action.

The Deterministic Rule (DR) is introduced as a heuristic baseline for determining when service replication should be performed. At each time instance t, the node observes the number of received requests Y_t and calculates the cumulative number of requests as $R_t = \sum_{k=0}^{t} Y_k$.

DR triggers service replication (pull-action) when either the cumulative number of requests exceeds the threshold θ or the deterministic time condition $t > \alpha \frac{\theta}{C}$ is satisfied. If neither condition has been satisfied when the final deadline T is reached, the node performs task offloading (push-action). Accordingly, the stopping time is defined as

$$t^* = \min\{0 \leq t < T : \sum_{k=0}^{t} Y_k > \theta \text{ or } t > \alpha \frac{\theta}{C}\}. \tag{2.5}$$

Both stopping conditions in Equation (2.5) trigger the pull-action. The push-action is triggered only when the final deadline T is reached without either stopping condition having been satisfied. The pseudo-code is provided in Algorithm 2.

Moreover, the proposed OST approach has been compared to the work in [64]. Their approach is based on choosing the stopping time using SECPRO model. The stopping time decision in [64] was based on choosing the highest value of the received requests per time instance:

$$Y^* = \max_t (Y_t), \tag{2.6}$$

by the edge node during the period $t \in \{0, \frac{T}{e}\}$, where e is Euler's number. Furthermore, taking into account calculating the cumulative sum of the number of the received tasks (R_t) and the cost C. Then, the node will resume receiving tasks during the period of $\{\frac{T}{e}, T\}$ where the first value $Y_t > Y^*$ is the preferred stopping time of this theory. If the first value is detected, it stops immediately. This time is considered the stopping time for this theory. Also, we would like to emphasize that during the period $\{\frac{T}{e}, T\}$ the cumulative sum of the received tasks (R_t) is calculated beside the value of the cost as shown in (2.7). The pseudo-code is provided in Algorithm 3.

$$t^* = \min\{\frac{T}{e} \leq t \leq T : \text{ such that } \sum_{k=0}^{t} Y_k \leq \theta \text{ and } Y_t > Y^*\} \tag{2.7}$$

As a result, the findings demonstrate the effectiveness of the proposed OST approach. This is what we will explain later in Section 2.3.3.

2.3.3 Sequential Decision Making based on Optimal Stopping Theory

2.3.3.1 Optimal stopping Rule Problem

The concept of optimal stopping [71] addresses the issue of selecting a time instance to conduct a specific action. The measure is taken to minimize an anticipated loss (or maximize an expected payoff). A stopping rule problem is characterized by: (i) a set of stochastic variables, whose joint distribution is presumed to be known, and (ii) a set of loss functions $(S_t(y_1, \ldots, y_t))_{1 \leq t}$ or payoff functions $(H_t(y_1, \ldots, y_t))_{1 \leq t}$ that rely exclusively on the observed values $y_1, \ldots, y_t$ of related random variables. An optimal stopping rule problem is expressed as follows: We are monitoring the series of $(Y_t)_{1 \leq t}$, and at each time instance t, we decide whether to stop monitoring or proceed. If we cease monitoring at instance t, we incur loss S_t or gain payoff H_t. We would like to pick a stopping rule or time that reduces our estimated loss or, equivalently, optimizes our expected payoff.

Definition 1: The optimal stopping time t^* minimises the expected incurred loss $E[S_{t^*}] = \inf_{0 \leq t \leq T} E[S_t]$. In the case of payoff, t^* maximises the expected payoff, i.e.,

$$E[H_{t^*}] \quad = \quad \sup_{0 \leq t \leq T} E[H_t]. \tag{2.8}$$

Note that T may be ∞. The information accessible up to time t is a series F_t of values of the random variables $Y_0, \ldots, Y_t$, which is also known as filtration.

Definition 2: The 1-stage look-ahead (1-SLA) stopping rule (time) is defined as:

$$t^* \quad = \quad \inf\{t \geq 0 : H_t \geq E[H_{t+1} | F_t]\}. \tag{2.9}$$

This means that t^* requires stopping at the first time instance t where the payoff H_t for stopping at t is as large as the anticipated payoff of proceeding to the next time instance $t+1$ and then stopping.

Definition 3: Let B_t denote the event

$$B_t = \{H_t \geq E[H_{t+1} \mid F_t]\}.$$

A stopping rule problem is monotonic if

$$B_0 \subseteq B_1 \subseteq \ldots$$

almost surely. The set B_t represents the set of states for which the 1-SLA rule requires stopping at time instance t. The condition $B_t \subseteq B_{t+1}$ indicates that if the 1-SLA rule requires stopping at time t, then it will also require stopping at time $t+1$, regardless of the value of Y_{t+1}. More generally, the condition

$$B_t \subseteq B_{t+1} \subseteq B_{t+2} \subseteq \ldots$$

states that once the 1-SLA rule requires stopping at time t, it will continue to require stopping at all future times, regardless of future observations.

Theorem 1. The 1-SLA rule provides the optimal solution for monotonic stopping problems.

Proof. See [72] □

The edge device is required to complete making the decision of service replication (pull-action) of the service of the number of tasks y_t that it has received. Because of this, the primary goal is to increase the payoff attached to the decision of service replication (pull-action). The edge node is actively monitoring a continuous sequence of received task requests. These tasks are being accumulated with those that have been received in the past with respect to performance criteria, which is denoted by the current delay cost C per time t. At each newly received task, the node should decide whether to make the decision of service replication (pull-action) or not. The challenge arises from the fact that the node wants to establish a service replication (pull-action) policy that will increase the chances of making the decision of service replication (pull-action) with a high payoff. As a result, the node needs to go for the optimal decision time that maximizes the payoff, which is globally rated as being the same as or near to the actual (ideal) payoff.

2.3.4 Cost-based Sequential Decision Making (COST)

Problem 1. *Given the tolerance threshold θ and delay cost $C > 0$, determine the optimal stopping time t^* such that* $\sup_{1 \leq t \leq \infty} E[H_t]$ *is attained. The greatest expected return is then* $E[H_{t^*}]$.

Problem 1 is addressed by establishing an OST-based model. Consider that the starting time $t = 0$ and the deadline is T, with $t < T$. Our OST model aims to stop receiving more tasks when the cumulative sum of the received task R_t is near to θ, taking into account the cumulative delay cost up to that point t. We provide a 1-SLA stopping rule based on the optimal stopping principle stated in Theorem 1, whereby we stop at the first time t such that:

$$H_t(R_t) \geq E[H_{t+1}(R_{t+1})|F_t], \text{ with the event } \{R_t \leq \theta\} \in F_t. \tag{2.10}$$

That is, any further received tasks at time $t+1$ would not contribute to maximizing the payoff. When the difference $E[H_{t+1}(R_{t+1})|F_t] - H_t(R_t)$ is constant non-increasing with R_t, the 1-SLA rule is optimal.

Theorem 2. Given a series of payoff random variables $R_1, \ldots, R_t$, the optimal stopping rule t^* for Problem 1 is provided in Equation (2.11).

$$t^* = \inf\{t \geq 0 : \sum_{y=0}^{\theta - R_t} (R_t + y) - C(t+1) \cdot P(Y = y) \leq R_t - C \cdot t\} \tag{2.11}$$

Proof. Given that $R_t \leq \theta$, the conditional expectation $E[H_{t+1}(R_{t+1})|R_t \leq \theta]$ is given by:

$$\begin{aligned} E[H_{t+1}(R_{t+1})|R_t \leq \theta] &= E_Y[H_{t+1}(R_t + Y)|R_t \leq \theta, R_t + Y \leq \theta]P(R_t + Y \leq \theta) \\ &+ E_Y[H_{t+1}(R_t + Y)|R_t \leq \theta, R_t + Y > \theta]P(R_t + Y > \theta) \\ &= E_Y[(R_t + Y) - C(t+1)|Y \leq \theta - R_t]P(Y \leq \theta - R_t) \\ &= \sum_{y=0}^{\theta - R_t} (R_t + y) - C(t+1) \cdot P(Y = y). \end{aligned}$$

Therefore, the mechanism terminates at the first time t, where R_t is such that $E[H_{t+1}(R_{t+1})|R_t \leq \theta] \leq R_t - C \cdot t$. □

The 1-SLA rule is optimal because it stops at the first time at which stopping yields at least as much expected payoff as continuing for one more step. In particular, when $R_t < \theta$, the rule compares the immediate payoff with the expected payoff of continuing, and stopping is triggered as soon as the former is no smaller than the latter. Under the monotonicity condition, once this inequality holds, it continues to hold at all later times. A high-level description of the provided Theorem 2 is that, it is always better to stop at the first time t rather than at $t+1$, because the current payoff at that t will be bigger than the payoff at the next time instance $t+1$, and any other time $t+\tau$ with $\tau > 1$. This is the principle of 1-SLA rule, where our problem falls into this category. Moreover, the difference of the payoff at the optimal time t and $t+1$ is always positive. However, that difference tends to decrease with the time t. This indicates that at the very first

Algorithm 1: Time-optimized Sequential Decision Making (OST)

Input: Tolerance threshold θ; observation cost $C > 0$
Output: Optimal stopping time t^*

```
Stop ← False
t ← 0
R_0 ← 0
repeat
    observe number of requests Y_t
    R_t ← R_{t-1} + Y_t        // update the cumulative sum of requests
    if criterion in (2.11) is satisfied then
        Stop ← True
        t* ← t      // activate service replication (pull-action)
         and start the new task service
        return t*
    else
        t ← t + 1          // continue with the next received task
until t > T
if Stop = False then
    trigger push-action
```

time t where the payoff is bigger than the future expected payoff, then it is always beneficial to stop at that t and not at any other future time, since the difference of the consecutive payoff values is always decreasing. Therefore, the 1-SLA rule in Theorem 1 is optimal; OST with fixed θ and taking into account the delay cost C may ensure that the anticipated cumulative sum of the received tasks based on the stopping criteria is as near as possible to θ, however, no other stopping rule can make such a claim. Based on the cost C, OST is adaptable for handling and controlling the latency and freshness of the service replication (pull-action) and offloading decision's variables. OST is concerned with the delay cost of monitoring and assumes that all received tasks are current for the application that uses the service replication (pull-action) and offloading decision. In this case, the OST is forced to terminate receiving new tasks to prevent waiting for an extended period of time, particularly when λ is very small. Therefore a large number of tasks are necessary to sum up to θ. The pseudo-code is provided in Algorithm 1.

2.4 Experimental Evaluation

2.4.1 Experimental Setup

In our experiments, we evaluate four different models: (i) the proposed OST model; (ii) the heuristic DR model, which depends on the heuristic factor α. In the DR model, according to Equation (2.5), the pull-action is triggered when either the cumulative number of requests satisfies $R_t = \sum_{k=0}^{t} Y_k > \theta$ or the deterministic time condition $t > \alpha\frac{\theta}{C}$ is satisfied. If neither

Algorithm 2: Heuristic decision-making model (DR)

Input: Tolerance threshold θ; observation cost $C > 0$; heuristic factor $\alpha \in [0,1]$
Output: Decided stopping time t^*

$Stop \leftarrow$ False
$t \leftarrow 0$
$R_0 \leftarrow 0$
while $t \leq T$ **do**
    observe number of requests Y_t
    $R_t \leftarrow R_{t-1} + Y_t$ `// update the cumulative sum of requests`
    if *criterion in (2.5) is satisfied* **then**
        $Stop \leftarrow$ True
        $t^* \leftarrow t$ `// trigger pull-action and start the new task service`
        break
    else
        $t \leftarrow t+1$ `// continue with the next received task`
if $Stop = False$ **then**
    trigger push-action

condition is satisfied before the final deadline T is reached, the push-action is triggered; (iii) the SECPRO approach introduced in [64]. SECPRO measures the current payoff when the decision is made using the OST criterion in (2.7); (iv) the IDL, which represents the real expected payoff (H) given in (2.4). This serves as the ground truth used to compare all the methods.

In the performance evaluation, we investigate the performance of the OST, DR, and IDL models. The parameters λ, α, and θ are varied across different experiments. Specifically, we set $\lambda \in \{3,10,30\}$ and $\theta \in \{50,200,600\}$. The experiments are conducted in three scenarios: the first experiment uses $\lambda = 3$ and $\theta = 50$, the second experiment uses $\lambda = 10$ and $\theta = 200$, and the third experiment uses $\lambda = 30$ and $\theta = 600$. These parameter values were selected to represent low, moderate, and high task arrival scenarios under the Poisson request model.

Furthermore, the observation cost C is assigned a range of values for each experiment. In particular, the cost values are defined as $C = (0.01, 0.05, 0.1, 0.3, 0.5, 0.9) \cdot \lambda$, which allows us to analyze different monitoring cost intensities relative to the request arrival rate. In addition, the heuristic factor α is varied across the range $(0.01, 0.05, 0.07, 0.09, 0.1, 0.3, 0.45, 0.6, 0.9)$ for all experiments, as shown in Table 2.2, in order to examine the sensitivity of the DR model under different decision aggressiveness levels.

Moreover, to obtain statistically reliable results, each experiment is repeated 1000 times, and the mean values of the obtained results are reported.

Algorithm 3: Secretary-based OST Model (SECPRO)

Input: Tolerance threshold θ; observation cost $C > 0$
Output: Secretary problem's optimal stopping time t^*

$Stop \leftarrow$ False
$t \leftarrow 0$
$R_0 \leftarrow 0$
$Y^* \leftarrow 0$
// observe task requests in the interval $[0, \frac{T}{e}]$
for $t = 0$ **to** $\frac{T}{e}$ **do**
    observe number of requests Y_t
    if $Y^* < Y_t$ **then**
        $Y^* \leftarrow Y_t$
    $R_t \leftarrow R_{t-1} + Y_t$ // update the cumulative sum of requests
// observe task requests in the interval $(\frac{T}{e}, T]$
for $t = \frac{T}{e} + 1$ **to** T **do**
    observe number of requests Y_t
    if $R_t > \theta$ **then**
        trigger push-action
    $R_t \leftarrow R_{t-1} + Y_t$ // update the cumulative sum of requests
    if *criterion in (2.7) is satisfied* **then**
        $Stop \leftarrow$ True
        $t^* \leftarrow t$ // activate service replication (pull-action) and start the new task service
        break
    else
        $t \leftarrow t + 1$ // continue with the next received task
if $Stop = False$ **then**
    trigger push-action

In our comparative assessment, we evaluated all four models, namely OST, DR, IDL, and SECPRO. The experiments were designed to analyse the behaviour of the models under different request arrival conditions and stopping thresholds. Specifically, the arrival rate was set to $\lambda \in \{3, 30\}$, the heuristic factor was fixed at $\alpha = 0.1$, and the stopping threshold was varied according to two different ranges: $\theta \in \{50, 100, 150, 200\}$ and $\theta \in \{600, 1200, 1800, 2400\}$.

The experiments were conducted in two stages. The first stage (Experiment 4) was configured with $\lambda = 3$, $\alpha = 0.1$, and $\theta \in \{50, 100, 150, 200\}$. This experiment aims to evaluate the behaviour of the competing models under relatively low request arrival rates and moderate stopping thresholds. The second stage (Experiment 5) was configured with $\lambda = 30$, $\alpha = 0.1$, and $\theta \in \{600, 1200, 1800, 2400\}$. This experiment aims to investigate the scalability and robustness of the models under significantly higher request arrival intensity and larger threshold values.

Table 2.1: Notations of parameters

NOTATIONS	Definition
$\mathcal{N}$	Set of EC nodes.
T	Deadline.
t	Time instance.
t^*	Optimal stopping time.
Y_t	Number of tasks received at time t.
λ	Arrival rate of tasks Y.
θ	Tolerance threshold.
C	Cost of the decision delay.
R_t	Accumulation of number of tasks Y_t.
α	Heuristic factor in DR model.
H_t	Payoff at time instance t.

Notation	Parameter	Value/Range
T	Deadline	100.
λ	Arrival rate of tasks	$\{3, 10, 30\}$.
θ	Tolerance threshold	$\{50, 100, 150, 200, 600, 1200, 1800, 2400\}$.
C	Delay cost	$\{0.01, 0.05, 0.1, 0.3, 0.5, 0.9\}$.
α	Heuristic factor	$\{0.01, 0.05, 0.07, 0.09, 0.1, 0.3, 0.45, 0.6, 0.9\}$.

Table 2.2: Experimental Parameter Setting.

Each experiment was repeated 1000 times, and the mean value of the results was reported. The objective of this comparative evaluation is to examine the overall behaviour of the models under different parameter settings rather than the variability of individual runs. Therefore, the mean value is used as the primary summary statistic, as it provides a stable and representative estimate of the expected performance over a large number of repetitions, enabling a consistent comparison between the evaluated models. A summary of the experimental settings and their objectives is provided in Table 2.3.

Experiment	Parameter Setting	Objective
Exp.1	$\lambda = 3$, $\theta = 50$	Evaluate model behaviour under low task arrival rate.
Exp.2	$\lambda = 10$, $\theta = 200$	Evaluate model behaviour under moderate task arrival rate.
Exp.3	$\lambda = 30$, $\theta = 600$	Evaluate model behaviour under higher task arrival rate.
Exp.4	$\lambda = 3$, $\alpha = 0.1$, $\theta \in \{50, 100, 150, 200\}$	Compare OST, DR, IDL, and SECPRO under low-demand conditions.
Exp.5	$\lambda = 30$, $\alpha = 0.1$, $\theta \in \{600, 1200, 1800, 2400\}$	Evaluate scalability and robustness under high arrival rate.

Table 2.3: Summary of experimental scenarios and objectives.

2.4.2 Performance Evaluation

All experiments follow the same evaluation setup. In each experiment, the parameters λ and θ define the request arrival intensity and the stopping threshold, while the performance is mainly influenced by the cost parameter C and the heuristic factor α. The experiments measure both the obtained payoff and the time at which the decision is made. Three models are evaluated throughout the experiments: the IDL model, which serves as the baseline reference, the proposed OST model, and the heuristic DR model. It should be noted that the SECPRO model was not included in Experiments 1–3. The purpose of these experiments was to evaluate the behaviour of the proposed OST approach against the baseline optimal reference (IDL) and the heuristic rule (DR) under different request arrival rates. Therefore, these experiments focus primarily on validating the effectiveness of the OST decision mechanism and its ability to approximate the optimal behaviour represented by IDL. The SECPRO method is introduced later in the comparative assessment experiments (Experiments 4 and 5), where the objective is to provide a broader comparison between different sequential decision-making approaches.

2.4.2.1 Experiment 1

The settings for Experiment 1 are $\lambda = 3$ and $\theta = 50$, representing a low service demand rate. Figure 2.5 illustrates the effect of the cost parameter on the three models and presents the corresponding payoff results. The results show that the proposed OST approach performs much closer to the IDL baseline than the DR model.

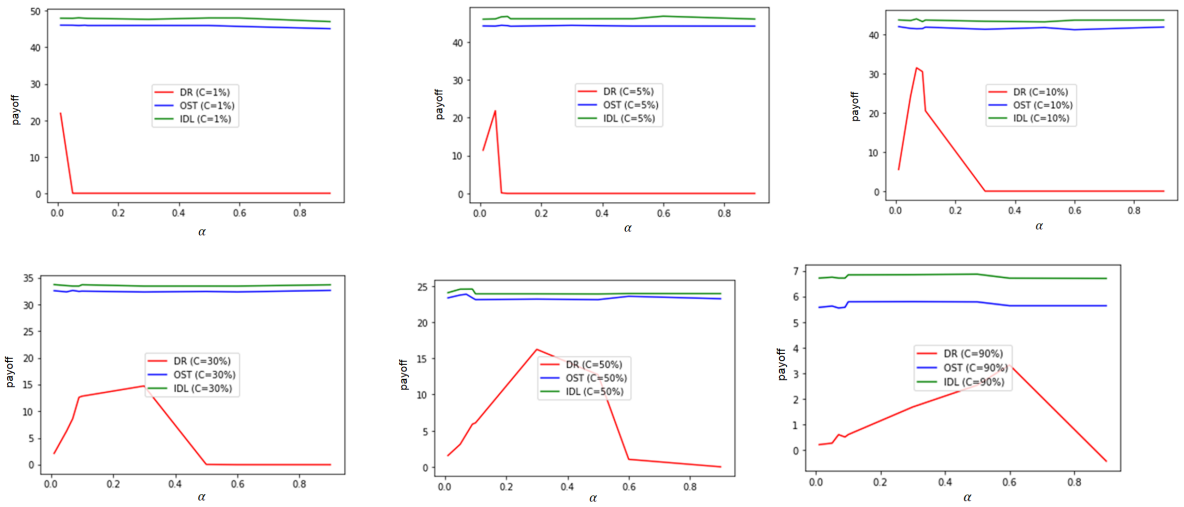


Figure 2.5: (Experiment 1) The expected payoff vs delay cost with low service demand rate $\lambda = 3$ and tolerance $\theta = 50$ for OST, DR and IDL models.

2.4.2.2 Experiment 2

The settings for Experiment 2 are $\lambda = 10$ and $\theta = 200$, corresponding to a moderate service demand rate. This experiment examines how the competing decision models behave under increased task arrival intensity compared with Experiment 1. Figure 2.6 illustrates the effect of the cost parameter on the three models and presents the corresponding payoff results.

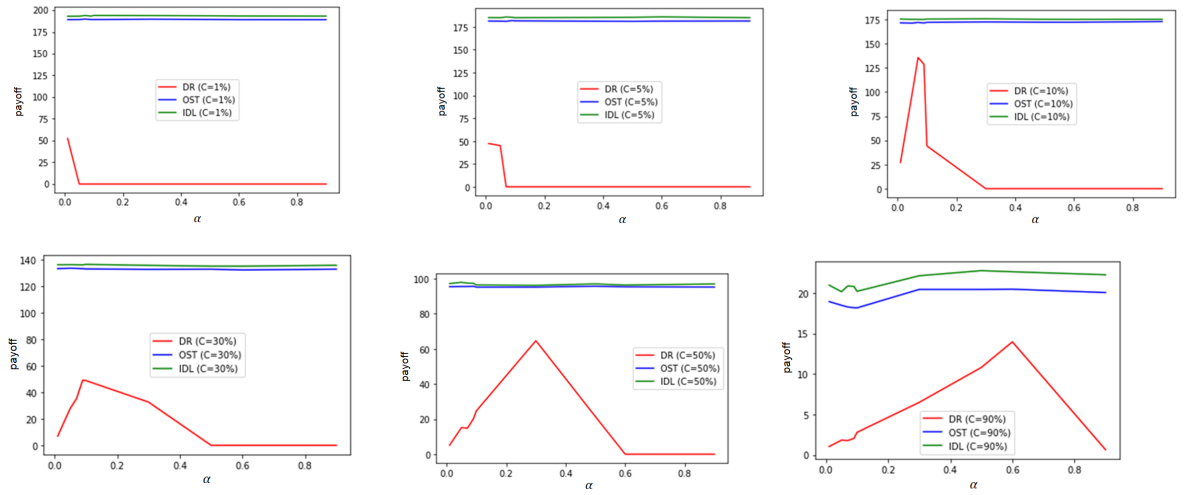


Figure 2.6: (Experiment 2) The expected payoff vs delay cost with medium service demand rate $\lambda = 10$ and tolerance $\theta = 200$ for OST, DR and IDL models.

2.4.2.3 Experiment 3

The settings for Experiment 3 are $\lambda = 30$ and $\theta = 600$, representing a high service demand rate. This experiment evaluates the behaviour of the models under substantially higher request intensity and larger stopping thresholds. Figure 2.7 illustrates the effect of the cost parameter on the three models and presents the corresponding payoff results.

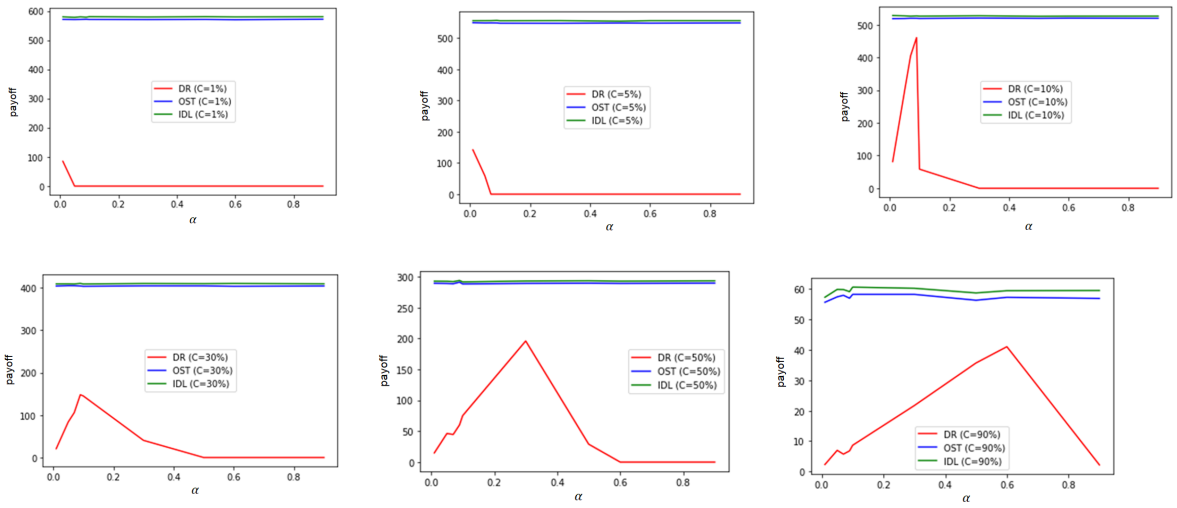


Figure 2.7: (Experiment 3) The expected payoff vs delay cost with high service demand rate $\lambda = 30$ and tolerance $\theta = 600$ for OST, DR and IDL models.

Figure 2.7 shows the different effects of the cost all over the three models and shows the result of payoff and how our approach is as close as IDL compared with DR.

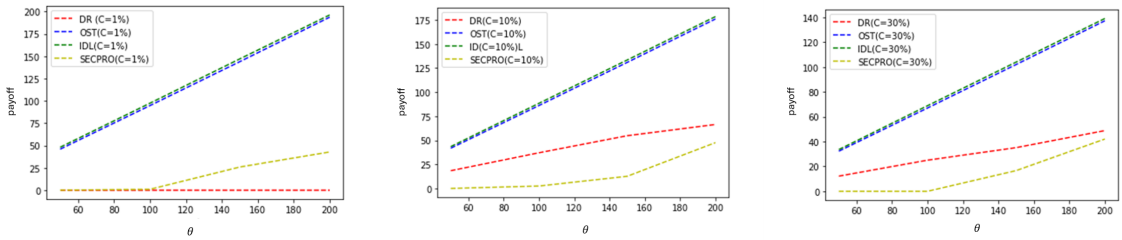


Figure 2.8: Expected payoff vs tolerance θ for diverse delay cost values with low service demand rate $\lambda = 3$ and tolerance $\theta \in \{50, 100, 150, 200\}$ for DR, OST, IDL and SECPRO models.

The results of Experiments 1–3 lead to several important observations. First, the proposed OST model consistently achieves performance that is very close to the IDL baseline, which represents the optimal expected payoff. Second, the heuristic DR model exhibits larger deviations from the IDL reference, indicating that heuristic-based decisions are less effective in balancing the monitoring cost and the accumulated number of task requests. Third, the performance of

the OST model remains stable as the arrival rate increases from low to high demand scenarios. These results demonstrate that the proposed OST-based framework provides a reliable and robust decision mechanism for determining the appropriate stopping time under different task demand conditions.

2.4.3 Comparative Assessment

The settings for the comparative assessment experiment 4 are: $\lambda = 3$, $\theta \in \{50, 100, 150, 200\}$ and $C = (1\%, 10\%, 30\%) \cdot \lambda$ (low service demand rate). We performed the experiment with the defined λ and $\alpha = 0.1$. Our experiment is mainly affected by different values of θ and C. Furthermore, the experiment measures the payoff V^* and the time t^* that decision has been made. Moreover, the experiment was performed over four models. The first model is IDL which is being chosen to be the baseline. It used to be compared with the other three models. The second model is our fundamental approach which is OST. Then, the third model is DR. Finally, the fourth model is SECPRO. The experiment tested four different models. Figure 2.8 shows the different effects of θ over all four models. Moreover, Figure 2.8 shows that the payoff achieved by OST is closer to the IDL baseline than the payoffs achieved by DR and SECPRO.

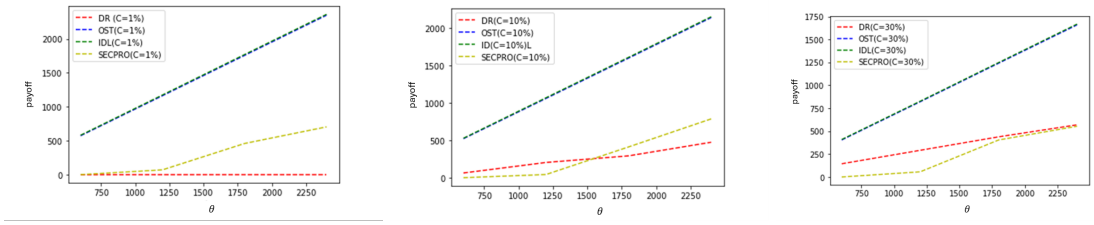


Figure 2.9: Expected payoff vs tolerance θ for diverse delay cost values with high service demand rate $\lambda = 30$ and tolerance $\theta \in \{600, 1200, 1800, 2400\}$ for DR, OST, IDL and SECPRO models.

The settings for the comparative assessment experiment 5 are: $\lambda = 30$, $\theta \in \{600, 1200, 1800, 2400\}$ and $C = (1\%, 10\%, 30\%) \cdot \lambda$ (high service demand rate): Here, we performed the experiment with the defined λ and $\alpha = 0.1$. Our experiment is mainly affected by different values of θ and C. Furthermore, the experiment measures the payoff V^* and the time t^* that decision has been made. Moreover, the experiment was performed over four models. The IDL model is used as the baseline and is compared with the other three models. The second model is our fundamental approach which is OST. Then, the third model is DR. Finally, the fourth model is SECPRO. The experiment tested four different models. Figure 2.9 shows the different effects of θ over all four models. Moreover, Figure 2.9 shows that the payoff achieved by OST is closer to the IDL baseline than those achieved by DR and SECPRO.

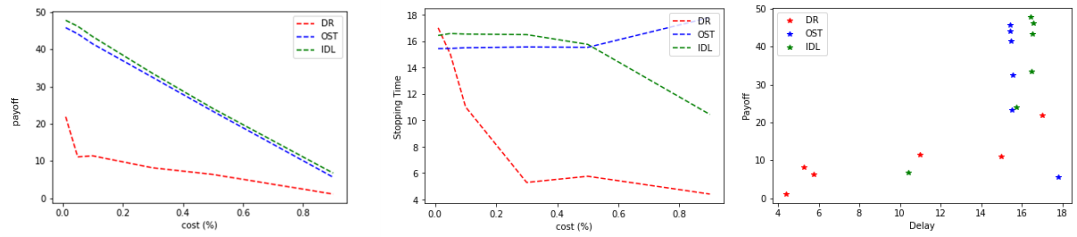


Figure 2.10: Expected payoff vs cost and delay for OST, IDL (low service demand rate $\lambda = 3$ and tolerance $\theta = 50$).

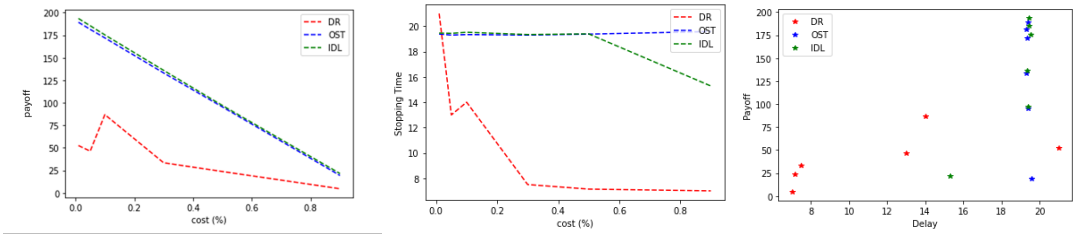


Figure 2.11: Expected payoff vs cost and delay for OST, IDL (medium service demand rate $\lambda = 10$ and tolerance $\theta = 200$).

The first three experiments evaluate the performance of the proposed OST method by comparing its results with IDL and DR. The models were tested under different parameter settings, ranging from low to moderate and high request-arrival intensities. The results indicate that the proposed OST method achieves performance close to the IDL baseline across the evaluated settings. As shown in Figures 2.10, 2.11, and 2.12, Figure 2.10 corresponds to the setting $\lambda = 3$ and $\theta = 50$, while Figure 2.11 corresponds to $\lambda = 10$ and $\theta = 200$. Similarly, Figure 2.12 corresponds to $\lambda = 30$ and $\theta = 600$. Each figure contains three graphs illustrating the relationships between cost and payoff, cost and stopping time, and delay versus payoff.

Figure 2.10 shows that the payoff achieved by OST is close to that of the IDL baseline in the cost-versus-payoff results. When the cost exceeds 0.5, a small difference appears in the cost-versus-stopping-time results. For cost values of 0.5 or less, OST achieves a shorter decision delay than IDL.

Figure 2.11 presents the results for $\lambda = 10$. The cost-versus-payoff graph shows close agreement between OST and the IDL baseline. For cost values above 0.5, the difference in decision delay decreases, while the stopping-time results of OST remain close to those of IDL.

Figure 2.12 presents the results for the high request-arrival setting, where $\lambda = 30$. The results show that OST maintains performance close to the IDL baseline when the cost exceeds 0.5, particularly in the cost-versus-stopping-time and delay-versus-payoff graphs. The cost-versus-payoff graph also shows close agreement between OST and IDL.

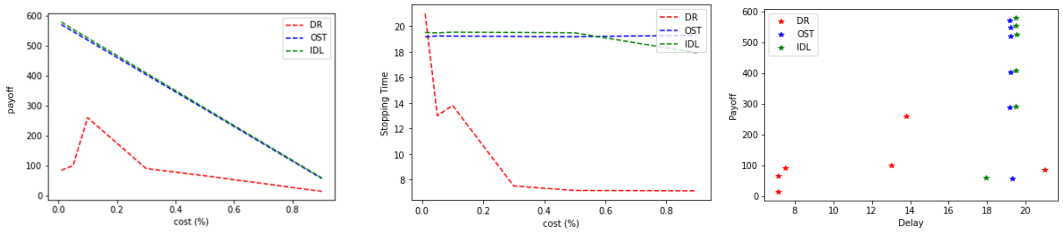


Figure 2.12: Expected payoff vs cost and delay for OST, IDL (high service demand rate $\lambda = 30$ and tolerance $\theta = 600$).

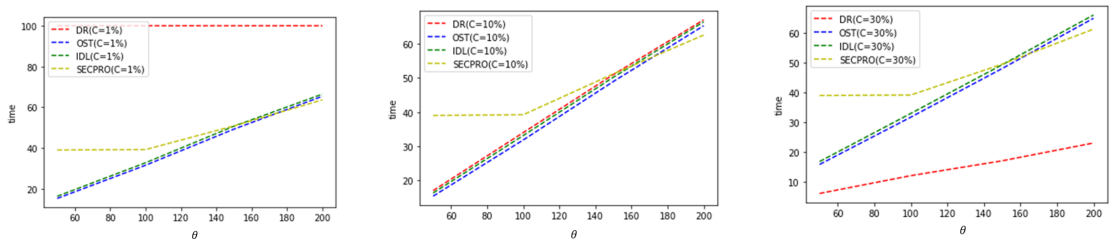


Figure 2.13: Expected delay (time to decision) vs tolerance θ for diverse cost values (low service demand rate $\lambda = 3$ and tolerance $\theta \in \{50, 100, 150, 200\}$).

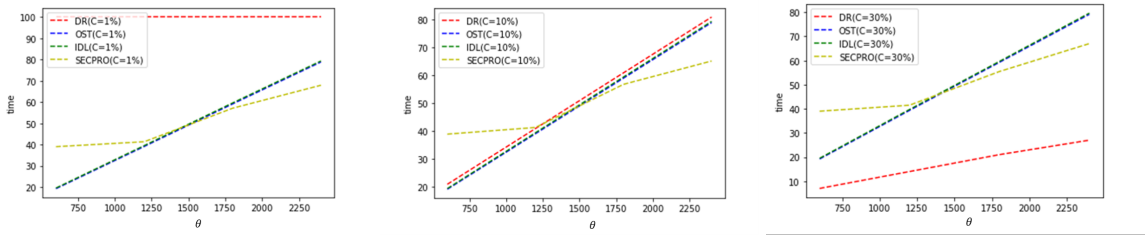


Figure 2.14: Expected delay (time to decision) vs tolerance θ for diverse cost values (high service demand rate $\lambda = 30$ and tolerance $\theta \in \{600, 1200, 1800, 2400\}$).

Moreover, Experiments 4 and 5 evaluate the performance of the proposed OST method by comparing it with IDL, DR, and SECPRO under low and high request-arrival settings. The results indicate that OST achieves performance close to the IDL baseline under the evaluated parameter settings. Figure 2.13 presents the results for $\lambda = 3$, $\theta \in \{50, 100, 150, 200\}$, and $C = (1\%, 10\%, 30\%) \cdot \lambda$, whereas Figure 2.14 presents the results for $\lambda = 30$, $\theta \in \{600, 1200, 1800, 2400\}$, and $C = (1\%, 10\%, 30\%) \cdot \lambda$. Each figure contains three graphs illustrating the relationship between the decision time t^* and the tolerance θ under different cost values.

2.5 Conclusions

We provide a superior decision-making approach for users in EC that is based on the Optimal Stopping Theory. When using and adopting the OST in time-optimized decision-making for a service replication (pull-action) in EC environments, we offer a comprehensive review of the process. Based on the results of our experimental assessments, the OST model performs more effectively than the other approaches, which are suitable for use in Edge nodes, and do not require a significant amount of resources. Moreover, the highest payoff is obtained by our models, which outperform alternative baseline solutions. In further work, our aim is to analyze how the characteristics of mobile applications and data timeliness affect our OST models and build new ones that take these limitations into account.

Chapter 3

Task Offloading in Mobile Edge Computing Using Cost-Based Discounted Optimal Stopping

3.1 Introduction

Advancements in the Internet of Things (IoT) have significantly influenced several application domains, such as smart homes, smart agriculture, manufacturing, and healthcare. To support these applications, a growing number of heterogeneous IoT devices are connected to provide real-time monitoring and actuation capabilities. These devices generate massive volumes of data that are typically transmitted to the cloud for processing, often referred to as big data. However, centralized processing in the cloud is unsuitable for many IoT applications for several reasons. First, latency-sensitive applications cannot tolerate the delays introduced by centralized cloud-based deployment [73]. Second, centralized architectures increase the likelihood of data loss and network failures [74]. Third, continuously uploading data to the cloud may significantly reduce the battery life of IoT devices [75]. Finally, some applications require tightly coupled request-response interactions [76].

To address these challenges, several computing paradigms have been proposed to bring cloud-like resources closer to end devices. In particular, mobile edge computing, fog computing, and cloudlets have attracted considerable attention. These approaches are commonly grouped under the broader concept of edge computing platforms. Mobile Edge Computing (MEC) represents a network architecture paradigm that places computing and storage resources at the edge of mobile networks [77]. By operating in close proximity to end users, MEC can provide a service environment with reduced latency, higher bandwidth, and direct access to real-time network information [78].

Mobile Cloud Computing (MCC) was initially proposed to support computation offloading by relying on mobile devices and centralized cloud servers [79]. However, since cloud servers are typically distant from mobile devices, offloading operations may introduce additional latency and communication overhead [80]. MEC was therefore introduced to mitigate the latency limitations associated with MCC by bringing computational capabilities closer to end users [81]. As a result, MEC can provide significantly lower transmission and computation delays compared with traditional cloud-based offloading approaches [82]. Despite these advantages, storage and processing capacities at MEC nodes remain limited. Nevertheless, MEC nodes can support essential computational services required to execute various processing tasks [35]. In many scenarios, such tasks are issued as service requests that must be processed by the available edge resources [83]. Moreover, nodes may offload tasks to neighboring peers by sharing services or data when necessary. However, edge nodes can host only a subset of the available services and data [27].

Consequently, a task may request a service that may or may not be locally available depending on the current service placement [35]. As discussed in [4], nodes have two possible actions in this situation:

1. Nodes may perform service replication (pull-action) from the cloud or neighboring edge peers when the service is required and compatible with their computational capabilities.

2. Nodes may offload the task (push-action) to peers or cloud resources that currently host the required service.

In the previous chapter, we applied Optimal Stopping Theory (OST) to regulate service replication decisions in an Edge Computing (EC) environment, enabling EC nodes to increase their operational autonomy. In this framework, each EC node determines locally when it is appropriate to replicate a service in order to support incoming tasks efficiently. Each node makes this decision independently according to the observed task requests and the node's current state. In this chapter, we extend this analysis by considering scenarios where the number of received tasks may vary, particularly when it decreases due to MEC mobility. In such situations, MEC nodes must determine individually when it is optimal to replicate a service in order to maximize the opportunity to execute incoming tasks efficiently. By applying the principles of Optimal Stopping Theory [84], the proposed approach enables MEC nodes to monitor service demand and determine the appropriate moment for service replication based on their local observations.

The remainder of this chapter is organized as follows. Before presenting the details of the proposed OST-based decision-making model in Section 3.3, we provide an overview of the related work, rationale, and novelty in Section 3.2. Section 3.4 presents the results of the performance and comparative evaluation, while Section 3.5 concludes the chapter and outlines possible directions for future work.

3.2 Related Work

3.2.1 Related Work

It is advisable to start comparing our current work with the previous work [4], as the strengths that encouraged us to embark on this work will be evident from the comparison. The work in [4] focused on decision-making in Edge Computing (EC) devices, which are mainly static. Through this, EC devices receive a fixed average number of requested tasks, and then a decision on service replication is made. In contrast, our current work carries the advantage of supporting MEC devices in making the appropriate pull-action decision for the required services. Moreover, in this study, we consider scenarios where the number of received task requests may decrease over time. This situation may occur due to the mobility of MEC nodes or users, which changes the set of nearby devices generating requests. In practice, the request rate may also decrease due to variations in workload patterns or service demand. As a result, MEC devices use the proposed method to make the most appropriate replication decisions individually based on the observed request dynamics.

In [85], the authors examine the possibility of offloading tasks in a dynamic MEC framework. They propose a hybrid energy supply strategy in which energy-collecting technology is included in IoT devices. In order to reduce the overall system cost, the authors coordinate the optimisation of local processing and offloading time. Moreover, they provide a randomised optimization-based online dynamic task offloading technique for MEC using a hybrid energy supply. This technique is able to offload tasks based on system cost and queue stability. However, our strategy is distinct from that one because our methodology focuses mainly on the actual advantage of the reward, which reflects the requests existing users have made for a particular service. Offloading decisions are basically formulated in [44], [45] on the basis of a resource scheduling factor. However, our technique determines whether to decide on a push-pull action based on the total number of newly assigned tasks and the associated costs.

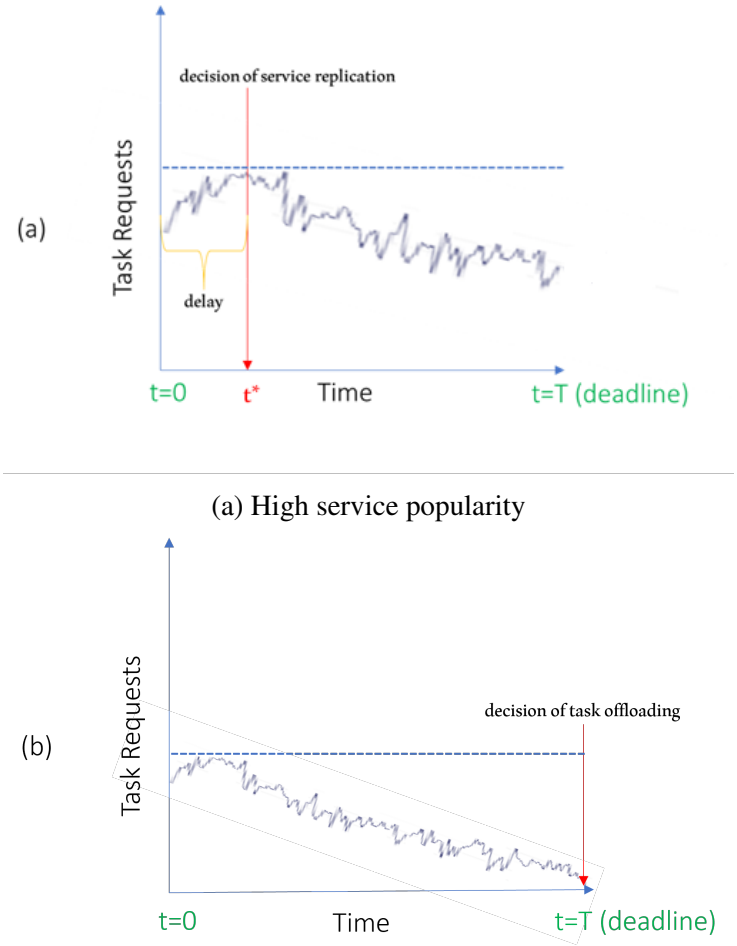
The work presented in [86] provides a task offloading strategy that takes into consideration both the categorization of tasks and the selection of offloading nodes. The objective of that study is to minimize the completion time of each task. However, their approach focuses on task offloading decisions based on task classification and node selection. In contrast, our work focuses on determining the appropriate decision for service replication based on the observed demand for services at the edge nodes. In [87], the authors offer a dependent task offloading architecture for multiple MEC environments. MEC devices may offload their compute-intensive duties within this framework to the MEC-Cloud system while maintaining dependent restrictions. In order to provide a better experience for the client, it is able to delegate offloaded tasks to the MEC and the cloud dynamically. Since the approach presented in the previous chapter handles limited-capacity scenarios more effectively than the approach in [87], our method focuses on determining whether to perform a pull or push action for a single user at an edge node or a peer.

The authors deal with offloading decisions in [88] based on energy and task causality limitations because of channel oscillations and the arrival of dynamic tasks over time. They do so by jointly improving the transmission power allocation at the energy transmitter (ET) for wireless power transfer (WPT) and the task allocation at the user for local computing and offloading over a particular finite horizon. This allows them to reduce the total transmission power used at the ET while maintaining the successful task execution of the user. This is accomplished over a particular finite horizon. However, the implementation of our solution may vary based on the level of demand for the service and the point of making the decision at the maximum reward. The approach in [46] copes with the problem of resource management at the edge server and as a means of determining the most effective offloading decision to lower overall system costs. In addition, the work occurring in [47] tries to make a choice to offload based on reducing costs to the absolute minimum. Our work, on the other hand, focuses on selecting the route that will result in the most significant reward while also taking into account the extra cost incurred if we skip over the option of push-action.

3.2.2 Rationale & Contribution

In Figure 3.1, we provide a simplified representation of the problem when the rate of incoming tasks decreases over time and a node must decide whether to replicate a service (i.e., pull-action) from its neighbours on the network edge or from the Cloud and handle the processing locally or to offload the task to the Cloud provider (i.e., push-action) and let it handle the processing. Suppose that the time when the number of task requests begins to decrease is $t = 0$, and that the task must be completed by a deadline $T > 0$. Two choices with limited time are shown in Figure 3.1. In this illustration, t denotes a discrete observation interval rather than an instantaneous point in continuous time. The variable Y_t represents a single aggregate value corresponding to the number of task requests observed during interval t. Therefore, each observation interval is associated with one request-count value, while the connecting line is used only to illustrate the temporal trend in service demand.

The situation shown in Figure 3.1 (a) is one in which the pull-action choice is made early in time t^* (and before the deadline T) due to the expectation of getting no deep decrease of received tasks seeking the specified service. This indicates that there is a sufficiently large number of tasks interested in this service, suggesting that pulling the service locally to the node is a good option. Moreover, one can observe what would have happened if we postponed making a decision with the help of our growing assurance that interest in the service is low, as shown in Figure 3.1 (b). Assuming that the service request rate was insufficient, the node has the option to offload the requests (push-action) to its peers or the Cloud after the deadline has passed without a pull-action being agreed upon. This means that offloading is required since there was a low amount of tasks seeking the specified service. Let us assume that the node made the decision to download (pull) the service early on, and then, after some time, we discovered that only a



(a) High service popularity

(b) Low service popularity

Figure 3.1: Illustration of the number of task requests observed over successive discrete time intervals. Each point represents one aggregate request-count value Y_t for interval t, while the connecting line illustrates the temporal trend. Subfigure (a) indicates high service popularity, whereas Subfigure (b) indicates low service popularity.

tiny percentage of users actually wanted it. If we had not performed the pull service action, we would have wasted more resources. Therefore, as shown in Figure 3.1 (b), it would have been beneficial to postpone our decision until the completion of the specified time frame and then carry out the offloading action. However, if the node had chosen to offload the request to the Cloud at an earlier moment, the node would not have had sufficient confidence that this request was frequently demanded at the node, which could lead to unnecessary communication overhead and increased latency. This service might still be requested in the near future. In such a situation, immediately adopting a push-based decision may lead to unnecessary offloading and increased communication delay. Therefore, the node should delay the decision until sufficient evidence about the service demand is observed, allowing a more informed choice between the pull-action and push-action alternatives.

An early push action may offload tasks to other peers or the Cloud prematurely, leading to unnecessary communication overhead if the node later determines that the service demand has not significantly decreased. Therefore, as illustrated in Figure 3.1(a), the node postpones the decision until a suitable stopping time t^* is reached. During the observation period $0 \leq t < t^*$, incoming requests are temporarily buffered at the node while the demand for the service is monitored. Once the decision time t^* is reached, the node performs a pull-action to replicate the service locally and then processes the buffered requests. Finding the optimal balance between task offloading and service replication in the scenario of receiving a decreased number of tasks is the challenge we face now that we know it has been formulated as t^*. Taking into consideration the latency costs, a creative solution has been provided that guarantees the right option is made at time t^*.

Contribution: Our methodology is flexible enough to be applied to applications that make use of offloading decision-making algorithms in MEC circumstances. The primary technological contribution that we make is:

- In the event that the number of tasks received decreases, we present a unique adaptive approach that increases the possibility of making the choice of service replication (pull-action) of the service with the highest (maximal) return. This is shown by a total cumulative of the total amount of task requests to the node that is being considered while taking notice of the cost that has been got.
- Using the concepts of OST, we theoretically analyse the superiority of our approach.
- We provide a comprehensive analysis of how our OST-based approach compares to our prior work [4], which generates the true optimum reward.

Figure 3.2 illustrates the decision-making process for determining whether to replicate a service (pull-action) or offload a task (push-action). The aggregate number of task requests observed during each discrete interval $t < T$ is a significant factor in this decision. The justification is as follows. During each discrete observation interval t, the node observes one aggregate request-count value Y_t. The node then has two choices:

- It needs to decide to perform service replication (pull-action) at a time t^* such that $0 < t^* \leq T$, if the number of requests is sufficiently large.
- Or, the node will re-evaluate its decision at time $t+1$, if there are incoming service requests.

If the deadline T is reached before a pull-action is triggered, the task(s) are offloaded (push-action) to a peer MEC node or to the cloud for further processing.

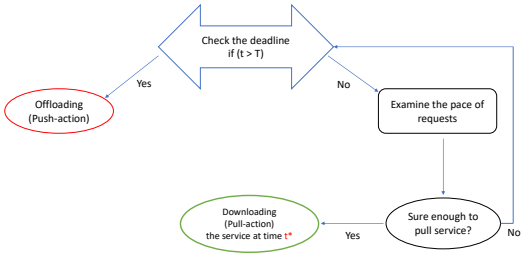


Figure 3.2: A diagram illustrating the sequential decision-making process for determining the optimal time $t^* < T$ for either task offloading (push-action) or service replication (pull-action) by the MEC node.

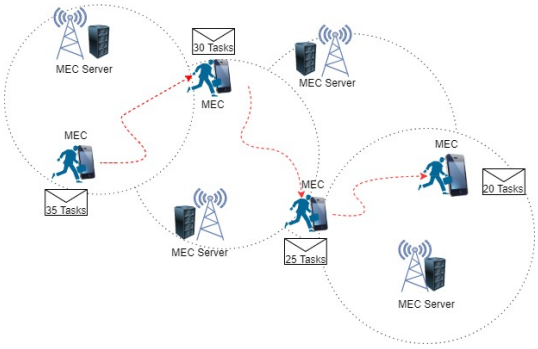


Figure 3.3: Example of MEC environment.

3.3 Problem Fundamentals

Firstly, we provide an overview of the system model and then in Section 3.3.2 we explore the description of the problem. In Section 3.3.3, we introduce our cost-based sequential decision-making with respect to maximizing the rate of return.

3.3.1 System Model

The considered network environment consists of a distant cloud data centre and a set of Mobile Edge Computing (MEC) base stations [67]. Figure 3.3 illustrates the MEC environment considered in this work. In the figure, each dotted circular region represents the coverage area of a MEC base station, which is equipped with a MEC server that provides computing and storage resources at the network edge. Mobile devices (illustrated in the figure by the user icons carrying smartphones) generate computational tasks that must be processed by the MEC infrastructure. These mobile devices are referred to as MEC devices in this work. The tasks generated by these devices can either be processed locally at the associated MEC server or offloaded to neighbouring MEC servers or to the distant cloud data centre when required [89]. Table 3.1 summarises

the notations used in this work. The set of MEC nodes is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$, where each MEC node corresponds to a MEC server deployed at a MEC base station. Multiple service providers may deploy their applications on these MEC nodes in order to serve incoming computational tasks.

Each MEC device generates task requests that are initially directed to its associated MEC node. If the required service is available locally, the MEC node can process the task directly. Otherwise, the node must decide whether to replicate the service locally (pull-action) from neighbouring MEC nodes or from the cloud, or to offload the task to another computing entity that already hosts the required service. Time is divided into discrete intervals denoted by $t = 1, 2, \ldots, T$. At a certain stopping time t^*, the MEC node makes a decision based on locally observed information, such as the number of received service requests. The decision determines whether service replication (pull-action) should be performed in order to process the tasks locally. The mobility of MEC devices may cause variations in the number of task requests received by a given MEC node. The red arrows shown in Figure 3.3 illustrate the possible movement of mobile devices between the coverage areas of different MEC base stations, which may lead to fluctuations in the observed service demand at each node. Consequently, the decision process must adapt to these variations in task arrival patterns.

Notation	Definition
$\mathcal{N}$	Set of MEC nodes.
T	Deadline.
t	Time instance.
t^*	Optimal stopping time.
Y_t	Number of tasks received at time t.
λ	Arrival rate of tasks Y.
θ	Tolerance threshold.
b	Cost of the decision delay for the DOST model.
c	Cost of the decision delay for COST model.
R_t	Cumulative sum of the number of tasks Y_t up to t.

Table 3.1: Notations of parameters

3.3.2 Problem Definition

The task model assumes that the number of requests for a particular task at a given time instant, denoted by Y_t, follows a Poisson distribution with rate λ [70]. Since MEC devices are mobile, the number of task requests observed by a given MEC node may vary over time as devices move between the coverage areas of different MEC base stations. To capture this behaviour, we as-

sume that the expected number of task requests decreases over time according to an exponential decay model. Specifically, the number of tasks received after time t follows a Poisson distribution with mean μq^{t-1}, where q is the decay factor ($0 < q < 1$) and μ represents the initial mean arrival rate.

Accordingly, the cumulative number of requests observed up to time t is used to evaluate the demand for the service and to support the decision-making process at the MEC node. We define the cumulative distribution function of the demand rate at t as:

$$P(Y <= y; \lambda) \quad = \quad \sum_{k=0}^{y} \frac{e^{-\mu} \cdot \mu^k q^{k(t-1)}}{k!} \tag{3.1}$$

Here, the parameter λ represents the overall arrival rate of task requests in the system. However, due to the mobility of MEC devices, the effective demand observed at a MEC node may decrease over time. Therefore, the parameter μq^{t-1} is used to model the time-varying mean demand at time t, where μ denotes the initial mean arrival rate and q^{t-1} captures the temporal decay in the observed demand.

Consider now the cumulative sum of the reduced number of requests R_t, $0 \leq t \leq T$ as:

$$R_t = \sum_{k=0}^{t} Y_k. \tag{3.2}$$

Problem 2. *Find a stopping rule t^* that maximises the expected rate of return per time instance t in (3.3), where $R_t = Y_0 + \ldots + Y_t$:*

$$t^* = \arg\max \frac{\mathbb{E}[R_t - b]}{t+1}. \tag{3.3}$$

Let $b > 0$ denote a fixed cost associated with the pull-action decision. Our objective is to determine the optimal stopping time t^* that maximises the expected rate of return defined in Equation (3.3).

The rationale behind Problem 2 lies in the fact that we expect the rate of service requests over time t, excluding the latency cost component, not to decrease significantly. Therefore, this exemplifies the possibility that the pull-action will prove to be stochastically a more advantageous option than the push-action would be. This is due to the fact that we have become more certain that in the (not-too-distant) future, we will be receiving a sufficient number of requests for a service in a node despite the fact that the number of requests is continuing to decrease. Because of this, our goal is to maximise this rate of return.

3.3.3 Cost-based Discounted Sequential Decision-making (DOST)

In order to discover a solution to our problem in maximizing (3.3), we first derive a stopping rule that maximises $\mathbb{E}[W_t - \lambda D_t]$, where $W_t = R_t - b$ and $D_t = t + 1$, with rate $\lambda > 0$. We first check whether the one-stop look-ahead policy (1-SLA) is the optimal one. Specifically, if we quit after a period of time t, the corresponding reward rate is:

$$R_t - b - \lambda(t+1) \tag{3.4}$$

Moreover, if we proceed to the next time instance $t+1$ and then stop, we would anticipate that we will have gained:

$$R_t + \mathbb{E}[Y_{t+1}] - b - \lambda(t+2) = R_t + \mu q^t - b - \lambda(t+2). \tag{3.5}$$

The difference between the reward obtained by stopping at time t and the expected reward obtained by continuing to time $t+1$ is

$$(R_t - b - \lambda(t+1)) - \big(R_t + \mu q^t - b - \lambda(t+2)\big) = \lambda - \mu q^t. \tag{3.6}$$

Since μq^t decreases monotonically with time t for $0 < q < 1$, the quantity $\lambda - \mu q^t$ increases monotonically with t. Therefore, there exists a first time at which stopping becomes preferable to continuing, and the optimal policy is the one-step look-ahead (1-SLA) rule. In particular, stopping becomes optimal when

$$\lambda \geq \mu q^t. \tag{3.7}$$

Solving this inequality for t gives

$$t \geq \frac{\log(\mu/\lambda)}{\log(1/q)}. \tag{3.8}$$

This leads to the following theorem.

Theorem 3. The OST rule at which a MEC node stops for the first time T_1 [72], and which maximizes (3.3), is given by:

$$T_1 = \min\{t > 0 : \lambda \geq \mu q^t\} = \min\left\{t > 0 : t \geq \frac{\log(\mu/\lambda)}{\log(1/q)}\right\}. \tag{3.9}$$

Proof. The 1-SLA policy in Theorem 3 is the optimal criterion that can maximize the rate of return. We proceed with estimating this time T_1 as follows. We can re-write the 1-SLA given the initial mean μ as [90]:

$$T_1 = m, \text{ where } \begin{cases} m = 1 & \text{if } \lambda \geq \mu \\ m = \frac{\log(\mu/\lambda)}{\log(1/q)} & \text{if } \lambda < \mu \end{cases} \tag{3.10}$$

The expected return anticipated from (3.10) is a function of λ:

$$\begin{aligned} V(\lambda) &= \mathbb{E}[R_m - b - \lambda(m+1)] = \\ &= \mu(1+q+\cdots+q^{m-1}) - b - \lambda(m+1) \\ &= \frac{\mu(1-q^m)}{(1-q)} - b - \lambda(m+1). \end{aligned} \tag{3.11}$$

To obtain the optimal value of λ corresponding to the stopping time m, we set $V(\lambda) = \mathbb{E}[R_m - b - \lambda(m+1)]$ equal to zero, since this condition represents the point at which the expected accumulated reward $\mathbb{E}[R_m - b]$ is balanced by the corresponding time-dependent term $\lambda(m+1)$. The resulting value of λ is then iteratively updated together with m, as described below, until convergence to the optimal solution is achieved.

$$\lambda = \frac{\frac{\mu(1-q^m)}{(1-q)} - b}{(m+1)}. \tag{3.12}$$

In practice, the parameters μ and q are treated as input quantities and can be estimated from historical observations of task arrivals before running the decision procedure. Based on a λ value from (3.12), we iteratively substitute it into (3.10) to obtain an updated value of $T_1 = m$, where m is initialized as a feasible positive integer with $m \geq 1$. This updated value is then substituted back into (3.12) to obtain a new value of λ. The recursion continues until convergence is achieved. The outcome is the optimal (maximal) expected rate of return λ along with the optimal stopping time m, where a MEC node should stop within the horizon $\{1, \ldots, T\}$ in order to maximize the rate of return in (3.3). □

The proposed cost-based decaying sequential decision-making procedure is summarized in Algorithm 4.

Algorithm 4: Cost-based Decaying Sequential Decision-making (DOST)

Input: Initial mean rate μ; decaying factor $0 < q < 1$; delaying cost $b > 0$
Output: Optimal rate and stopping time t^*

```
Stop ← False
t ← 0
R_0 ← 0
repeat
    observe the number of requests Y_t
    R_t ← R_{t-1} + Y_t        // update the cumulative sum of requests
    if criterion in (3.9) is satisfied then
        Stop ← True
        t* ← t     // activate service replication (pull-action)
         and start the new task service
        return t*
    else
        t ← t + 1          // continue with the next received task
until t > T
if Stop = False then
    trigger push-action
```

3.4 Experimental Evaluation

3.4.1 Experimental Setup

We test two distinct models in our experiments that show evidence of our approach: (i) The DOST model, which is determined by the initial rate μ; $0 < q < 1$ incurred cost $b > 0$. In our proposed model, the decision to perform a pull-action is made if the accumulation value, R_t, at the time instance, t, meets the criteria outlined in (3.10). (ii) The COST model in [4], which calculates the cumulative total of the received tasks R_t based on the stopping criterion is as close as feasible to a budget threshold θ. After this, it makes the choice of which service replication (pull-action) will result in the biggest reward while also taking into consideration the cost that will be incurred c for each time instance t, as shown in (3.13). In every other case, the decision to offload is taken if the deadline elapses.

In the experiments that were designed to provide a comparative analysis in Section 3.4.2, we conducted the test using both the DOST and the COST models. In addition, we determine the values for each of the variables μ, q, and θ, as well as the cost b and cost-per-time c. Furthermore, we adjusted $\mu = 10$, $c = 0.1\mu$, and $b = 0.1\mu$. Moreover, we split the experiments into two groups. The first group included an unchanged value for budget $\theta = 50$. After that, we conducted the comparative tests with respect to the payoff of the DOST and COST for $q \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$. Finally, we analysed the results of the experiments. Furthermore, our assessment involves fixing a value for $q \in \{0.7, 0.8, 0.9\}$ and running comparison

experiments between the payoff of DOST and COST for various values of $\theta \in \{10, 30, 50, 100\}$. All parameters are defined to simulate different scenarios that examine the models in different restrictions, for instance, capacity that is denoted by θ and decay factor q. In the second group, we carried out experiments where we compared the amount of time required to stop the DOST and COST models along with the amount of time required to stop with the associated payoff.

3.4.2 Comparative Assessment

Choosing the moment to stop and maximizing cumulative reward using OST is the foundation of the strategy proposed in [4], which we term here as Cost-based Sequential Decision Making (COST). The goal is to prevent more tasks from being accepted when the cumulative total of those tasks, denoted by the variable R_t, is getting close to a value θ, which represents the tolerance threshold (budget). In addition to that, taking into consideration the total amount of incurred cost $c > 0$ due to delays up to that moment t. As a result, the optimal stopping rule with the highest payoff achieved in [4] is provided in (3.13) for reasons of completion:

$$t^* = \inf\{t \geq 0 : \sum_{y=0}^{\theta - R_t} (R_t + y) - C(t+1) \cdot P(Y = y) \leq R_t - C \cdot t\} \tag{3.13}$$

The COST model is compared with our proposed DOST approach. The many effects that q has on both models in terms of payoff are shown in Figure 3.4. Also, from Figure 3.4, we notice that there are different values for the q, which means that $q = 0.5$, there is a sharp decline in the number of tasks received, and as the value of q increases, the decrease in the number of tasks received decreases. In this scenario, we notice the red line, which represents the value of payoff across the different q values of the COST model. On the other hand, we also notice the blue line, which represents the value of the payoff for our new approach, the DOST model. Furthermore, it is noted that the DOST model has a high return on payoff compared to our previous work COST, regardless of the severity of the decline in the number of requests received. However, our previous work was clearly affected by the sharp drop in the number of tasks received, as it shows a low value of payoff compared to our DOST model, and these payoffs improve as the value of q increases. Despite that and on all q values, the effectiveness of the performance of our DOST model has a higher return of payoff than our previous work COST model, and this demonstrates the strength of the principle that we have chosen to work with the challenge of decreasing the number of tasks within the specified time T. It should be noted that the DOST formulation assumes $0 < q < 1$. Hence, the rightmost point in the figure represents a value of q approaching 1, not the exact case $q = 1$. Therefore, the results at that point should be understood as representing the near-limiting behaviour of the model rather than the exact behaviour at $q = 1$.

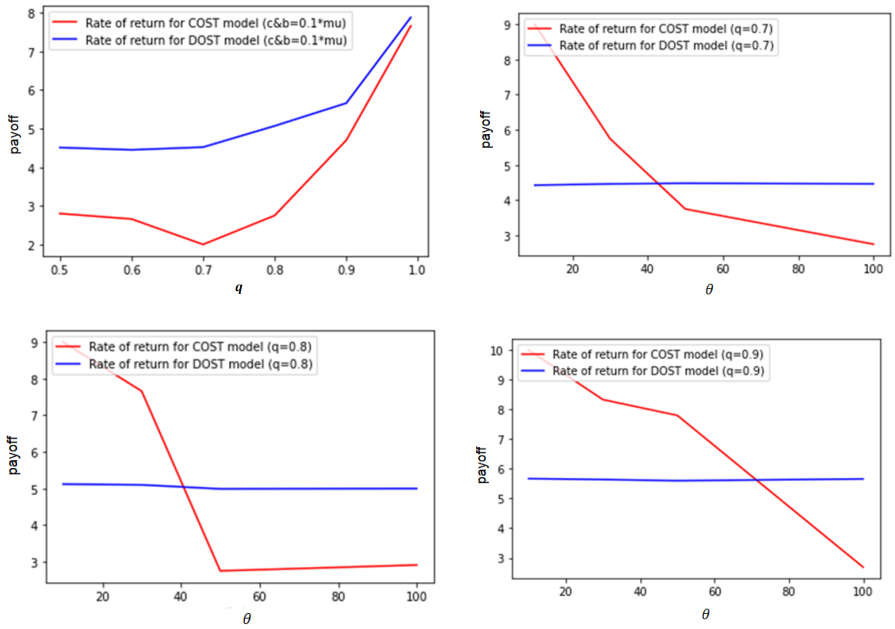


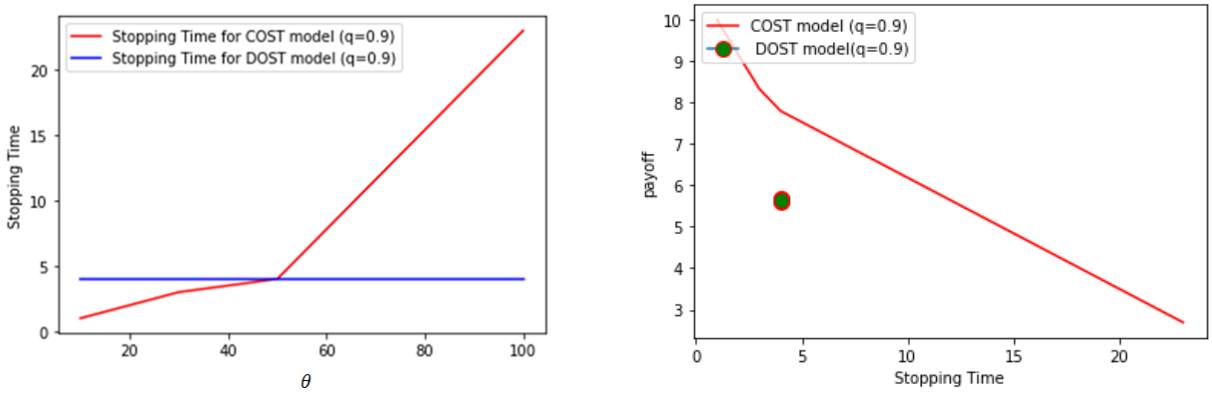
Figure 3.4: (i) Expected payoff vs q for fixed delay cost value $c\&b = 0.1 * \mu$ for COST and DOST models; (ii) Expected payoff vs θ for fixed $q = 0.7$ value for COST and DOST models; (iii) Expected payoff vs θ for fixed $q = 0.8$ value for COST and DOST models; (iv) Expected payoff vs θ for fixed $q = 0.9$ value for COST and DOST models.

For all comparisons shown in Figure 3.4, the common parameters were set to $\mu = 10$, $c = 0.1\mu$, and $b = 0.1\mu$. In the first plot, θ was fixed at 50, while $q \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$ was varied. In the second, third, and fourth plots, $\theta \in \{10, 30, 50, 100\}$ was varied, while q was fixed at 0.7, 0.8, and 0.9, respectively. For $q = 0.7$ and $q = 0.8$, the DOST model maintains a relatively stable payoff as θ increases, whereas the payoff of the COST model decreases more noticeably. The DOST payoff at $q = 0.8$ is higher than that observed at $q = 0.7$, reflecting the slower decay in the number of received tasks. At smaller values of θ, COST may be comparable to or outperform DOST; however, as θ increases, DOST generally achieves a higher and more stable payoff.

Furthermore, we set $q = 0.9$, as shown in Figure 3.4. In this case, the red line represents the COST model and the blue line represents the DOST model. The results indicate that the DOST model preserves a stable payoff over a wide range of θ values, whereas the COST model exhibits a decreasing payoff as θ increases. Since $q = 0.9$ corresponds to a relatively mild decay in the number of received tasks, both models achieve better payoff values at smaller values of θ. Even so, the proposed DOST model consistently outperforms the earlier COST model for most values of θ, particularly in scenarios involving MEC devices with greater service capacity.

The parameters used for the second group of the comparative assessment experiment are $\mu = 10$, $c = 0.1\mu$, and $b = 0.1\mu$. In this group, the decay factor was fixed at $q = 0.9$, while the threshold θ was varied over the range $\{10, 30, 50, 100\}$. The experiment evaluates the stopping time t^* at which the highest expected payoff is achieved. Figure 3.5a presents the relationship between stopping time and θ for the COST and DOST models. The red line corresponds to the COST model, while the blue line represents the DOST model. The results show that the stopping time of the COST model varies significantly as θ increases.

In contrast, Figure 3.5b illustrates the relationship between stopping time and expected payoff for the two models. The results indicate that the DOST model maintains a relatively stable stopping time across different values of θ, whereas the COST model tends to delay the stopping decision as θ increases. Since $q = 0.9$ corresponds to a mild decay in the number of received tasks, both models achieve favourable payoff values at smaller values of θ. Nevertheless, the proposed DOST model consistently achieves better performance than the earlier COST model, particularly in scenarios involving MEC devices with greater service capacity.



(a) Stopping time vs θ.

(b) Stopping time vs expected payoff.

Figure 3.5: Performance comparison of DOST and COST models for $q = 0.9$.

3.5 Conclusions

We propose a time-optimized decision-making strategy for MEC systems based on Optimal Stopping Theory (OST) to maximise the rate of return. This decision-making technique is particularly useful in situations in which users receive tasks with variable demand rates. We provide a full evaluation of the procedure when it comes to utilizing and implementing the DOST mechanism for service replication (pull-action) in MEC systems. According to the findings of our experimental evaluations, the performance of DOST outperforms COST, a related OST-based approach. These results are appropriate for use in MEC nodes and are not influenced by the total amount of resources that are available. Furthermore, the best results are achieved by our model, which is superior to baseline solutions in terms of payoff.

In the future, our goal is to develop a new model that adopts AI-driven techniques for selecting the best set of MEC nodes to offload the tasks relying on multi-armed bandits algorithms.

Chapter 4

Gradient Bandit Experts For Node Selection In Edge Computing

4.1 Introduction

The node selection issue in edge computing entails determining the most appropriate edge nodes for task offloading, taking into account their constrained computational capacity, memory, and storage. Various tasks exhibit distinct requirements, making node selection a broad process with a significant impact on performance. The heterogeneity of hardware, software, and network connectivity further contributes to the selection challenge, as these elements influence task execution effectiveness. Although selecting robust nodes in proximity to the data source is typically advantageous, it is not always practical due to resource constraints and variable network conditions. Additionally, the dynamic characteristics of edge environments result in fast alterations in computational and network infrastructure, impacting node accessibility and reliability. Consequently, optimizing node selection is vital to ensure effective task execution and maintain the overall performance of the edge computing ecosystem.

Different machine learning methods have been devised for node selection in edge computing [91–99]. Nonetheless, the majority assumes a static environment, ignoring real-time feedback signals from edge nodes. In applications such as autonomous vehicles, healthcare monitoring, and industrial automation, rapid and accurate node selection is crucial for prompt processing and responses. Several systems are capable of gathering real-time feedback, including response time and computational load, to evaluate node conditions. Despite this, the majority of current methodologies depend on predetermined configurations. In dynamic situations, integrating real-time feedback is essential for enhancing decision-making and assuring optimal node selection in edge computing systems.

This study presents a multi-armed bandit (MAB)-based technique for node selection in edge computing, enhancing the Gradient Bandit algorithm with expert guidance. To improve decision-making, we incorporate autoregressive deep learning models trained on historical data as expert predictors. MAB is a feedback-oriented sequential learning framework that balances exploration and exploitation to improve real-time decision-making [100, 101]. In the context of node selection, each action corresponds to selecting a candidate edge node, while the reward reflects the observed performance of that node after executing the assigned task, such as response time, resource utilization, or successful task completion. MAB-based approaches have been applied in several domains, including node selection in edge computing [98, 99], web-based news recommendation systems [102], and online advertising [103]. In this chapter, our approach is compared with existing node-selection methods in edge environments. Classical MAB algorithms typically assume stochastic environments; however, edge computing systems often exhibit non-stationary behaviour where node performance may change over time due to variations in workload, network conditions, or resource availability. Consequently, the node-selection problem can resemble an adversarial or dynamically changing setting in which node performance may depend on prior system states. While this chapter focuses on contextual bandit experiments, Chapter 5 extends the analysis by considering delayed and missing feedback conditions.

The remainder of this chapter is organized as follows. Section 4.2 reviews the related work and outlines the main contributions. Section 4.3 presents the proposed node selection algorithm. Section 4.4 reports the experimental results. Finally, Section 4.5 concludes the chapter.

4.2 Related Work

Node selection for service placement and task offloading is a well-studied problem within the edge computing and related domains.

Lately, numerous machine learning-based models have been suggested for node selection in edge computing [91–94]. The vast majority of these methodologies examine various architectural and data division techniques to enhance the conventional supervised learning paradigm, hence improving performance metrics such as latency and immediate response time. Given the continuous communication between edge devices and edge nodes via various feedback and heartbeat signals, it is rational to store and train a model utilizing this characteristic data. The system can initially commence the collection of diverse datasets from edge devices and edge nodes, including node-specific and task-specific data regarding CPU and memory utilization, energy consumption metrics, and network information. The past information can be refined, annotated, and utilized for training with cutting-edge machine learning techniques such as Deep Learning [91]. In [92], the authors examine computation division algorithms that leverage both node and device resources to enhance latency, energy efficiency, and data center throughput in

intelligent applications. The research, encompassing resource-intensive applications in computer vision, speech, and natural language processing utilizing cutting-edge Deep Neural Networks (DNNs), demonstrates that a fine-grained, layer-level division strategy provides substantial benefits over the conventional cloud-only processing method regarding latency and energy efficiency. [93] introduced Distributed Deep Neural Networks (DDNNs) intended for deployment within a distributed computing framework that includes nodes and end devices. DDNNs facilitate localized and efficient inference by executing shallow segments of the network at the edge and end devices while permitting more extensive processing in the cloud. The decentralized architecture of DDNNs improves sensor fusion, system fault tolerance, and data privacy. Through the mapping and concurrent training of neural network segments inside this hierarchy, DDNNs reduce communication expenses and resource consumption while enhancing feature extraction efficiency. The suggested DDNNs can utilize geographical sensor variety to enhance object detection accuracy and substantially decrease transmission costs. [94] presented a machine learning-based energy consumption model specifically designed to address energy utilization in edge computing during service placement. The authors offer an innovative offloading technique utilizing a Self-adaptive Particle Swarm Optimization algorithm integrated with Genetic Algorithm operators. This method improves the decision-making process for DNN layer offloading by optimizing layer division, hence decreasing the computational burden and enhancing execution performance. [104] suggested a two-stage machine learning-based approach for node selection aimed at real-time task offloading in highly dynamic edge computing environments for autonomous driving. The program initially forecasts the success or failure of work offloading through a classification model, subsequently employing a regression model to estimate service time. The methodology exhibits substantial enhancements in task failure probability and service durations by efficiently distributing computational weighs across edge and cloud resources, rendering it exceptionally useful in addressing the challenges of edge computing in autonomous vehicle systems. These works fundamentally contrast with ours, as these models are trained using historical data and lack consideration for the dynamic nature and rapid feedback signals from the environment.

More recently, bandit and reinforcement learning methodologies have been utilized for node selection in edge computing [95–99, 105–107]. For example, [105] introduced a deep Q-network to acquire optimal offloading strategies that reduce system latency and energy usage. This system adeptly allocates bandwidth and takes edge nodes to optimize performance in general. [106] examined the particular scenario of sub-task offloading in a game-theoretic structure, framing the issue as a multi-agent partially observable Markov decision process. They suggested a decentralized methodology utilizing deep reinforcement learning through policy gradient and differential neural network approaches. Likewise, [107] presented a deep reinforcement learning methodology for the online placement of services and allocation of computational resources in edge computing contexts. Their approach employs a parameterized deep Q-network

to concurrently optimize service placement and resource allocation, aiming to minimize overall task latency. Although these methods utilize feedback signals for training, they fail to account for quick feedback. The training data comprises feedback signals from past interactions among nodes, devices, and environments. In the Multi-Armed Bandit (MAB) framework, [108] introduced an MAB-based edge reputation management system for identifying the optimal node for data trading in autonomous vehicle management. A significant difficulty in privacy-preserving autonomous vehicles is the prohibition on uploading sensitive and confidential user data collected by onboard sensors to edge servers. The research examined encrypted information trading; in contrast to open-access data trading, encrypted data trading mitigates confidentiality and security issues while motivating contributors to share their knowledge. The suggested reputation management system assists edge servers in managing reliable and accessible vehicles, while it incurs a cost in data exchange. The MAB methodology enables edge servers to identify automobiles with excellent ratings for data exchange. A strategy of encapsulation is utilized to maintain the privacy and rights of the edge server while attaining the required degree of communication security.

In [98], the authors introduced multiple enhancements to established bandit algorithms, including the renowned Exp3 algorithm. Furthermore, [99] enhanced the widely-used bandit algorithm, Thompson Sampling, to address the multi-interface channel allocation issue against uncertainty. This study especially addresses the issue of bandwidth allocation across devices when an edge node communicates with numerous edge devices. Nevertheless, the authors considered binary feedback signals from the environment, which may not accurately reflect real-world conditions. Our problem setting markedly diverges from that of other bandit-related studies, which generally consider a stochastic environment and binary feedback signals. Conversely, our methodology is engineered to be robust in non-stochastic settings.

Notation	Definition
$\mathcal{N}$	Set of edge nodes.
T	Time horizon.
$r_{i,t}$	Reward of node n_i at time t.
U_t	Cumulative reward up to time t.
p_i	Selection probability of node n_i.
h_i	Preference value of node n_i.
ε_t	Noise term.
$R(T)$	Regret after T rounds.
s_t	1 if optimal node selected at t, else 0.

Table 4.1: Notations of parameters

4.3 Preliminaries & Algorithm

The MAB problem is a single-state reinforcement learning issue wherein a finite array of resources must be allocated among rival alternatives to optimize anticipated rewards. In the context of EC node selection, these possibilities signify the accessible nodes, presenting the issue as a bandit game between a decision-making algorithm (the node selection mechanism) and the environment. The environment illustrates the dynamic and frequently non-random characteristics of the EC nature.

Assume a discrete time domain $t \in \{1,2,\ldots,T\}$ with a limited horizon $T > 0$, within an EC ecosystem comprising $\mathcal{N}$ nodes. Upon the generation of a new task by an EC application, the node selection process is activated to determine the most appropriate node for execution within the specified time frame T.

Let $r_{i,t}$ denote the reward acquired by a node $n_i \in \mathcal{N}$ at time t. Furthermore, let $I_t \in \mathcal{N}$ denote the node chosen by the node selection process at time t. The objective of the method is to choose nodes in a manner that maximizes the cumulative reward U_t up to time t over the selected nodes $\{I_1,\ldots,I_t\}$, i.e.,

$$U_t = \max \sum_{\tau=1}^{t} r_{I_\tau,\tau}. \tag{4.1}$$

4.3.1 Evaluating Rewards in Edge Computing Configuration

Reward estimation is essential to any MAB-based algorithm, directing the selection process to prioritize high-reward actions. In conventional MAB configurations, rewards are generally binary feedback from the environment, signifying success or failure. Nonetheless, in EC node selection, such binary feedback is impractical. In EC node selection, continuous-valued feedback is employed as rewards rather than binary signals. We presume that this input reflects the existing condition of the node, including the quantity of active tasks and response time. In practice, such information can be obtained through periodic node-status updates, which we refer to here as heartbeat signals [109, 110].

4.3.2 ExpGradBand Algorithm for Node Selection

The ExpGradBand algorithm is proposed as an initial hybrid approach that integrates gradient-based adaptation with expert-weighted selection. The pseudo-code for the suggested **Exp**ert based **Grad**ient **Band**it (ExpGradBand) is presented in Algorithm 5. It utilizes a preference-based probability distribution to choose nodes, assigning weights $\boldsymbol{w}$ to experts according to their predictive ability. The preference matrix H represents the tendency of each expert to re-

commend a particular node. Based on these preferences, a probability distribution over the nodes is constructed using the softmax function. The algorithm first selects an expert according to the expert weights and then applies the Gradient Bandit mechanism to select a node and update the preferences according to the observed reward r_t.

The algorithm commences by initializing the preference matrix H for each expert–node pair, the expert weights $\boldsymbol{w}$, the average rewards $\bar{\mathbf{r}}$, and the counters $\boldsymbol{n}_i$ for each node. At each time step, it gathers recommendations from the expert set $\mathscr{E}$, computes their mean value, and normalizes the expert weights. An expert e is then sampled according to the weight distribution $\boldsymbol{w}$. Using the preferences of the selected expert, a probability distribution $\mathbf{P}$ over the nodes is computed, and a node I_t is selected from this distribution. The reward r_t obtained from the selected node is then used to update the average reward estimate and the preference values according to the gradient bandit update rule. Experts whose recommendations align better with the observed rewards gradually receive higher weights, allowing the algorithm to improve node selection over time.

The final three steps update the expert weights. First, θ forms a probability-adjusted feedback signal using the mean expert advice $\boldsymbol{v}$ and the node-selection probabilities $\mathbf{P}_{e,:}$ of the selected expert. The inner product $\theta' = \langle \mathbf{E}, \theta \rangle$ then measures how closely each expert's recommendation aligns with this signal. Finally, the multiplicative update $\boldsymbol{w} = \boldsymbol{w} \exp(\eta \theta')$ assigns relatively larger weights to better-aligned experts, increasing their probability of being selected in subsequent iterations. Since $\mathbf{P}_{e,:}$ is adapted through the reward-based Gradient Bandit update, the expert-weight update interacts indirectly with the observed reward.

Algorithm 5: Experts with Gradient Bandits (ExpGradBand)

Parameters: Set of Experts $\mathscr{E}$, Number of Nodes k, Time Horizon T, learning rate α, expert update rate η

Initialize : $H_{i,j} = 0 \quad \forall\, i = 1 \dots |\mathscr{E}|, \forall\, j = 1 \dots k$

Initialize : $\boldsymbol{w}_i = 1 \quad \forall\, i = 1 \dots |\mathscr{E}|$

Initialize : $\bar{\mathbf{r}}_i = 0 \quad \forall\, i = 1 \dots k$

Initialize : $\boldsymbol{n}_i = 0 \quad \forall\, i = 1 \dots k$

for $t = 1, 2, \dots T$ **do**

- $\mathbf{E} =$ get expert advice() `// collect recommendations from experts`
- $\boldsymbol{v} = \text{mean}(\mathbf{E})$
- $\boldsymbol{w}_i = \frac{\boldsymbol{w}_i}{\sum \boldsymbol{w}_i}$
- $e = \text{sample}(\boldsymbol{w})$ `// select an expert according to weights`
- $\mathbf{P}_{i,j} = \frac{\exp(H_{i,j})}{\sum_{l=1}^{k} \exp(H_{i,l})} \quad \forall\, i = 1 \dots |\mathscr{E}|, \forall\, j = 1 \dots k$
- $I_t = \text{sample}(\mathbf{P}_{e,:})$ `// select node using probability vector of expert` e
- observe reward r_t from I_t
- $\boldsymbol{n}_{I_t} \leftarrow \boldsymbol{n}_{I_t} + 1$
- $\bar{\mathbf{r}}_{I_t} = \bar{\mathbf{r}}_{I_t} + \frac{1}{\boldsymbol{n}_{I_t}} (r_t - \bar{\mathbf{r}}_{I_t})$
- **for** $i = 1 \dots k$ **do**
  - **if** $i = I_t$ **then**
    - $H_{e,i} = H_{e,i} + \alpha (r_t - \bar{r}_{I_t})(1 - \mathbf{P}_{e,i})$
  - **else**
    - $H_{:,i} = H_{:,i} - \alpha * (r_i - \bar{r}_i) * \mathbf{P}_{:,i}$
- $\theta = 1 - \frac{1 - \boldsymbol{v}}{\eta + \mathbf{P}_{e,:}}$ `// construct a probability-adjusted feedback signal`
- $\theta' = \langle \mathbf{E}, \theta \rangle$ `// compute an alignment score for each expert`
- $\boldsymbol{w} = \boldsymbol{w} \exp(\eta\, \theta')$ `// increase the relative weights of better-aligned experts`

Method Pipeline Overview

The proposed ExpGradBand framework follows a structured decision-making pipeline. First, a set of expert predictors generates performance estimates based on historical contextual feedback, such as resource utilization and heartbeat signals. Second, an expert selection mechanism identifies the most reliable predictor according to past prediction accuracy. Third, the selected expert's advice is integrated into a gradient bandit update rule, which adjusts node preferences

in an online manner. Finally, a decision step selects the target edge node for task execution. This pipeline enables ExpGradBand to combine predictive intelligence with lightweight online learning, allowing fast adaptation to changing edge conditions. The overall decision-making process is illustrated in Figure 4.1.

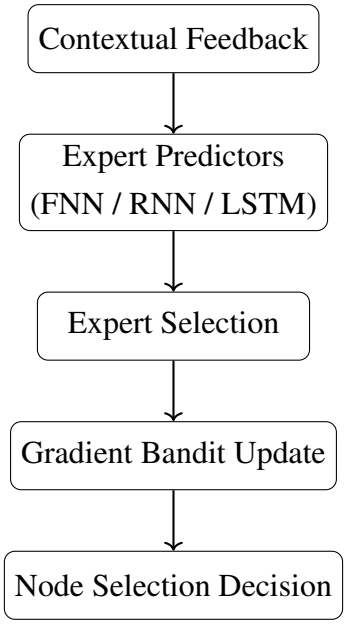


Figure 4.1: Decision-making pipeline of the ExpGradBand framework.

4.4 Experiments

Our purpose in this section is to evaluate the performance of the proposed ExpGradBand in comparison to the most advanced MAB-based node selection algorithms currently available.

We produce synthetic non-stochastic data for experimental purposes. Given that system states progress as time series affected by preceding states, we represent them by a basic Autoregressive (AR) process. AR models forecast future values utilizing historical observations, proposing that past data holds significant predictive insights. The fundamental concept of the autoregressive model is that the present value of a time series is linearly dependent on its preceding values (lags). This model proposes that historical values of the series include significant information regarding future behavior, therefore enabling predictions of future points from the analysis of past data. An autoregressive model is technically denoted as $AR(q)$, where q indicates the quantity of lagged observations incorporated in the model. In a formal manner:

$$z_t = c + \phi_1 z_{t-1} + \phi_2 z_{t-2} + \cdots + \phi_q z_{t-q} + \varepsilon_t, \tag{4.2}$$

Table 4.2: Hyperparameters of the edge nodes

#	ϕ_1	ϕ_2	**mean**
1	0.25	-0.25	$\exp(1.50)$
2	-0.45	0.25	$\exp(0.50)$
3	-0.35	0.40	$\exp(0.55)$

where z_t represents the current value of the time series, and $z_{t-1}, z_{t-2}, \ldots, z_{t-q}$ denote the q preceding values (lags). The constant term c represents the intercept, whereas $\phi_1, \phi_2, \ldots, \phi_q$ denote the coefficients (weights) that quantify the influence of each lagged value on the current value. ε_t denotes the error term, which encapsulates the randomness or noise that remains unexplained by preceding values.

In order to produce the data for the research we are conducting, we make use of a straightforward AR(2) model. We model a scenario with three edge nodes and six interconnected devices. At each time interval, the nodes transmit heartbeat signals that encode their current condition. We propose that the heartbeat signals convey the system state value such that higher values indicate more favorable nodes for selection. The various parameters for the nodes are presented in Table 4.2

4.4.1 Baselines

We use the following four baseline algorithms to compare our proposed approach.

Thompson Sampling (TS) TS is a lightweight and comparatively straightforward MAB algorithm grounded in the Bayesian principle. TS regulates the exploration-exploitation trade-off by sustaining a probability distribution for each node's prospective reward and then sampling from these distributions to guide decision-making. The prior distribution is continually revised according to observed rewards. In our research, we employed non-negative rewards in accordance with the formulas presented in [111].

Gradient Bandits The Gradient Bandit (GB) evaluates the preference value of each node, which affects selection, by utilizing feedback rewards [112]. Considering that the observed rewards of the presently selected nodes fluctuate, it is clear that the associated preference value likewise varies. A GB algorithm is designed to prioritize nodes with higher rewards over those with lesser rewards. This prioritization is accomplished by progressively modifying particular rules according to the observed rewards and the current likelihood of selecting the designated node. The concept of GB dictates that nodes with elevated preference ratings possess a greater likelihood of being selected.

Exponential-weight for Exploration & Exploitation (Exp3) Exp3 is designed to address situations where the distribution of rewards is variable and may fluctuate over time [113, 114]. In our scenario, the node selection mechanism utilizing Exp3 monitors the probability distribution of all nodes. This distribution is utilized for selecting the most suitable node at time t. The probability of selecting each node is modified based on the rewards acquired from the chosen nodes up to time t. Nodes that previously received relatively high rewards are awarded more weights, impacting future selection decisions. The weight of each node is adjusted based on the observed reward in an exponential fashion.

Exponential-weight Exploration and Exploitation using Expert advice (Exp4) The Exp4 algorithm operates by integrating recommendations from various experts. The method initiates by allocating uniform weights to all experts, presuming that their reliability is equivalent. In each round, the experts offer advice in the shape of probability distributions across the nodes. The system integrates these recommendations into a weighted probability distribution and selects the node based on this aggregated advice. The program adjusts the experts' weights subsequent to reward observation. Experts providing excellent advice, resulting in substantial benefits, experience an increase in their weights, whereas those proposing ineffective activities face a reduction in their weights [113].

4.4.2 Expert Oracles

In our analyses, we employ deep learning-based time series predictors as experts. We employed three distinct deep-learning models: *(i)* Feedforward Neural Network, *(ii)* Recurrent Neural Network, and *(iii)* Long Short-Term Memory Network.

4.4.3 Protocol & Metrics

We provide a data stream that reflects 30 minutes of operation, with heartbeat signals transmitted every second. The initial 15 minutes of data are utilized for training experts, while the last 15 minutes are employed to evaluate the performance of all algorithms. The accuracy of various algorithms is assessed using per-round regret analysis.

In the multi-armed bandit literature, regret measures the performance loss of a learning algorithm compared to the optimal decision that could have been made with full knowledge of the environment [100, 101]. In other words, regret quantifies how much reward is missed due to imperfect decisions during the learning process.

After T rounds, the per-round regret of an algorithm is defined as follows:

$$R(T) = \frac{T - \sum_{t=1}^{T} s_t}{T}. \tag{4.3}$$

Here, s_t is a binary variable that equals one if the optimal node is chosen at round t, and zero otherwise.

4.4.4 Performance Evaluation & Discussion

The results of our study are presented in Figure 4.2. The Exp4 algorithm exhibited lower performance relative to non-contextual bandit algorithms. While it exceeded Exp3, its total performance was markedly worse than that of GradBand and ExpGradBand.

Figure 4.2 illustrates that the cumulative regret of Exp4 (blue line) stabilizes at a higher level compared to the other approaches except for EXP3, signifying its ineffectiveness in mitigating regret. Conversely, ExpGradBand (cyan line) attains the minimal cumulative regret, illustrating its enhanced flexibility. GradBand (magenta line) and Thompson sampling (green line) surpass Exp4, demonstrating quicker convergence. Exp3 (black line) exhibits the poorest performance, underscoring its weaknesses in the present scenario.

Exp4's subpar performance arises from two principal issues. Reward scaling concerns impeded its learning process, as the method presumes rewards fall within the $[0,1]$ range. In our configuration, rewards occasionally surpassed this threshold, resulting in incorrect weight updates that impeded optimal node selection. This misalignment impeded the elimination of regret, rendering Exp4 less competitive. Furthermore, the selection of experts and the precision of predictions were essential. Exp4 allocates weights to experts according to historical accuracy; nonetheless, our results indicated that the LSTM-based expert consistently predicted system states accurately across all nodes. The uniform precision resulted in an equitable probability distribution among nodes, leading Exp4 to commonly choose subpar nodes. The system failed to distinguish between different node characteristics, resulting in heightened cumulative regret.

ExpGradBand achieves the lowest regret among the evaluated approaches because it can dynamically modify expert contributions while enhancing node selection. This adaptive learning technique enables quick decision refinement, resulting in expedited regret minimization. GradBand demonstrates effective performance, indicating that gradient-based updates alone can substantially improve selection techniques.

Thompson sampling initially varies but ultimately stabilises at a reduced regret level, illustrating its efficient exploration-exploitation balance. Exp4 encounters difficulties with expert-based weighting; nonetheless, Exp3, which avoids expert reliance, performs much more poorly due to its absence of contextual adaptation. Its regret decreases at a much slower rate, reaffirming its inefficiency in this scenario.

These results underscore the benefits of combining expert instruction with adaptive learning. ExpGradBand and GradBand substantially surpass conventional approaches, with ExpGradBand achieving the best performance among the evaluated approaches. The findings indicate that the efficient integration of knowledge from experts enhances decision-making efficiency, especially in dynamic contexts requiring adaptive learning. These results serve as preliminary evidence of the algorithm's potential. Next, Chapter 5 will examine its robustness under delayed, lost, and asynchronous feedback conditions across heterogeneous edge nodes.

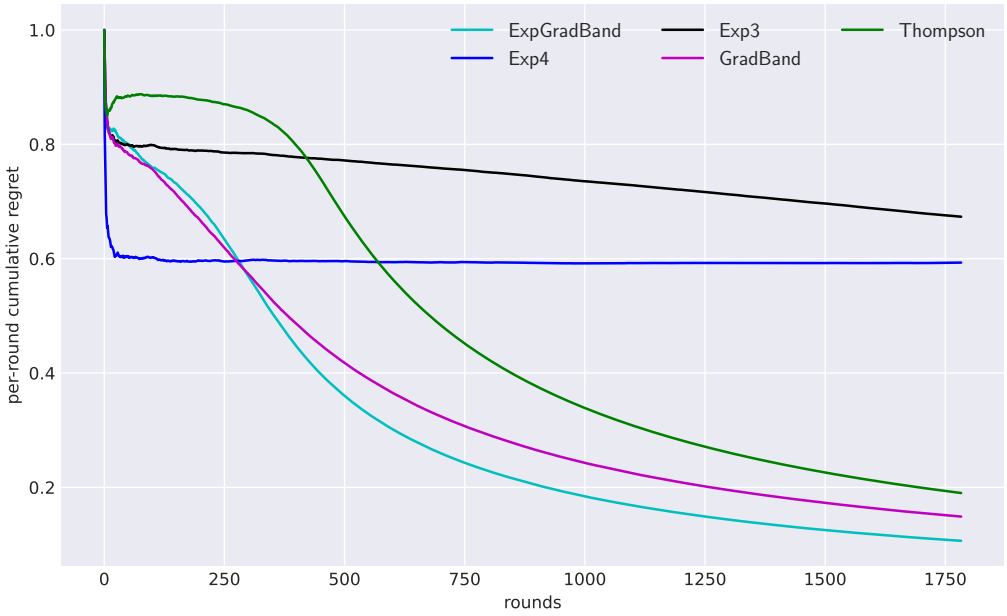


Figure 4.2: Experiments with Contextual Data over all MAB methods.

4.5 Conclusions

By formulating the node selection problem as a Multi-Armed Bandit (MAB) task, this chapter presented an enhanced decision-making strategy for edge computing environments. The proposed ExpGradBand methodology combines bandit-based exploration and exploitation with expert guidance to improve task offloading decisions. The experimental results indicate that ExpGradBand performs better than the considered baseline methods in terms of efficiency and resource utilization. These results suggest that the improvement arises from the interaction between adaptive bandit learning and expert-based prediction, rather than from any single component alone. Further analysis of the contribution of individual expert models will be presented in Chapter 5, where the evaluation is extended through additional experimental settings and performance metrics.

Chapter 5

Node Selection using Adversarial Expert-based Multi-armed Bandits in Distributed Computing

5.1 Introduction

5.1.1 Motivation

Edge Computing (EC) is a distributed computing paradigm that brings computation closer to data sources, such as sensors, Internet of Things (IoT) devices, and micro data servers, rather than relying on centralized data centers or cloud [115]. In the EC ecosystem, distributed edge nodes, hereinafter referred to as *nodes*, are dispersed at the network boundary, providing immediate collection of data from devices and exploration via local computing and interaction with other nodes. EC improves the quality of service and user experience by reducing response times, saving bandwidth, enhancing security, and ensuring reliability and privacy through data processing at the network edge. By processing data closer to where they are generated, EC significantly reduces latency, making it an ideal environment for applications requiring real-time processing, such as autonomous vehicles, augmented reality, and latency-sensitive smart-home services, including security surveillance, intrusion detection, and emergency response systems. EC offers flexibility in scaling by leveraging nodes to adapt to demand fluctuations while evenly distributing the computational load.

Recent developments in EC focus on improving node selection strategies for offloading *tasks* to nodes for execution. Nodes are used for (near) real-time events and data-driven computations required by, e.g., autonomous vehicles, wearable cognitive assistants, and IoT applications. In such scenarios, end-users demand highly efficient services with stringent time constraints, necessitating that tasks be completed within time limits. However, EC infrastructures are prone to failures, and a poorly designed infrastructure greatly impacts performance and service reliability.

This poses significant challenges during the prompt selection of the most appropriate nodes for, e.g., task offloading, delegation of tasks, or service requests [116]. Different tasks have different requirements for resources like computational power, memory, storage, and latency. This makes the selection of the most appropriate node a non-trivial decision-making process with significant implications for downstream operations. Moreover, the inherent heterogeneity of nodes in hardware, software, and network connectivity impacts the performance of tasks and consequently the decision of node selection. For instance, while selecting computationally powerful nodes that are physically closer to data sources can often be advantageous, it is not always feasible due to, e.g., node unavailability or overloading of node capacity. Therefore, the *node selection problem* becomes a crucial factor that directly impacts the overall efficiency of an EC environment.

5.1.2 Node Selection in Distributed Computing Environments

Before elaborating on node selection in EC, we discuss several application domains where EC infrastructure employs node selection mechanisms. Consider an EC environment comprising distributed edge devices (e.g., smart home appliances and autonomous vehicles) and edge nodes such as micro-data servers, base stations, or roadside units that host edge computing capabilities. Consistent with the definition introduced earlier in this thesis, edge nodes refer to resource-rich computing entities deployed near end devices to process tasks and support latency-sensitive applications. Therefore, computationally intensive tasks are executed through collaboration among devices and nodes [6]. In **smart home environments**, smart TV applications offer a seamless experience by streaming high-definition content and providing real-time interactive features, e.g., interactive gaming. Smart home infrastructure employs node selection mechanisms to choose the most appropriate node for offloading real-time video processing and analytics, ensuring optimal performance and user experience. In such settings, smart TVs typically act as end-user devices, while computational tasks may be dynamically offloaded to nearby edge resources, such as home gateways, local micro-servers, or ISP edge nodes that provide additional processing capacity. In addition, **smart climate control** systems in large-scale office buildings use a network of sensors and actuators to maintain optimal temperature, humidity, and air quality. Such systems leverage EC to locally process data, enabling real-time decision-making while ensuring a comfortable and energy-efficient environment. The climate control system selects the most suitable node for data processing, analytics, and communication with weather forecasting nodes in the smart city. The primary goal of node selection is to minimize decision-making latency. In practice, selecting nodes with lower response times often indirectly contributes to distributing workload across available nodes and maintaining reliable system operation.

Smart healthcare systems deployed in large-scale hospital units support healthcare services like remote patient monitoring, real-time diagnostics, and robotic-assisted surgery. Such systems rely on nodes distributed across hospital units to locally process and analyze data, ensuring low-latency and high-reliability services crucial for patient care (e.g., real-time monitoring, treatment planning, coordinated care, and predictive analytics). The smart healthcare system selects the most appropriate node for processing different types of medical data (e.g., vital signs), processing medical imaging data, or supporting robotic surgery.

Moreover, **autonomous vehicles** are equipped with various sensors (e.g., LiDAR, cameras, and radars) that generate massive amounts of data. Such data need to be processed in real-time for tasks like object detection, obstacle avoidance, and lane-keeping. These vehicles rely on a network of roadside nodes/units (RSU) that process real-time data, such as sensor inputs, traffic conditions, and vehicle-to-vehicle (V2V) communications. The goal is to ensure safe, efficient, and low-latency decision-making for tasks like navigation, collision avoidance, and traffic management.

5.1.3 Node Selection Strategies

Node selection methods are essential and widely used across various applications beyond EC to e.g., optimize resource allocation, enhance system performance, and improve efficiency. In Wireless Sensor Networks (WSNs), such methods are introduced to select the most appropriate cluster of sensor nodes for data aggregation [117]. In Peer-to-Peer (P2P) networking, node selection methods are used for engaging the most appropriate peers for relatively fast and reliable data transfer [118]. Additionally, node selection methods in social networking are used for influence maximization and community detection [119].

In the EC context, strategies for tackling the node selection problem can be broadly classified into the following categories: (i) heuristic methods [120], (ii) optimization-based methods adopting evolutionary algorithms and linear and/or integer programming [121], (iii) supervised learning methods [91], and (iv) reinforcement learning-based methods [97]. In our context, EC deals with real-time applications like healthcare monitoring and industrial automation, where prompt and accurate node selection is essential for timely processing and responses. In many of these scenarios, applications can obtain feedback signals that provide insights into the state (context) of each node, such as response time, computational load, and other relevant metrics. Most of the node selection mechanisms found in the literature are designed for static environments. Such approaches typically rely on a fixed, predefined EC setup, employing methods like heuristic and optimization algorithms to obtain static solutions or building predictive models based on historical data, as seen in supervised learning approaches.

In dynamic EC settings, where the context of edge nodes is time-varying due to changes in workload, response time, connectivity, and resource availability, a node selection mechanism based on real-time decision-making is deemed necessary along with incorporating immediate feedback signals from the nodes to be taken into account in selection decisions.

Under this context, Multi-armed Bandits (MAB), introduced earlier in Chapter 4, provide a suitable framework for feedback-driven sequential decision-making in node selection problems. MAB addresses the trade-off between *exploration* and *exploitation* when selecting actions (nodes) based on observed feedback signals [101]. In this setting, actions correspond to selecting candidate nodes, while rewards correspond to feedback signals derived from node observations (e.g., contextual information such as node load or connectivity latency).

MAB algorithms have been applied in various domains including node selection in EC, web-based recommendation systems, and online advertising [103]. Classical MAB formulations often assume stochastic environments where system performance follows an unknown probability distribution [98]. However, EC environments are frequently non-stochastic and dynamic. Node performance may depend on recent workload conditions, connectivity variations, or unexpected system events such as node failures or resource throttling [122, 123]. Therefore, node selection mechanisms must adapt to contextual feedback and evolving system conditions.

In this chapter, we investigate the applicability of the MAB framework for node selection in EC environments and study its behavior under challenging operational conditions where contextual feedback may be delayed or partially lost. Building on the ExpGradBand algorithm introduced in Chapter 4, we evaluate its effectiveness and compare it with state-of-the-art bandit algorithms for node selection under different feedback configurations.

The remainder of this chapter is organized as follows: Section 5.2 provides an overview of related work and reports on our contribution. In Section 5.3, we introduce the real-time node selection as a bandit problem discussing different bandit algorithms. Based on our experimental results, Sections 5.4 and 5.5 present the application and evaluation of the previously introduced **Experts with Gradient Bandits (ExpGradBand)** algorithm under different operational conditions. Furthermore, we compare ExpGradBand with state-of-the-art bandit algorithms for node selection found in the literature in different configurations including delayed and lost contextual feedback. Finally, Section 5.6 concludes the chapter with future directions of research in this area.

5.2 Related Work & Contribution

While a broader discussion of node selection literature was provided in Chapter 4, particularly in Section 4.2, this section presents a concise summary of the most relevant approaches for the purposes of this chapter. This section examines the main methodologies addressing the node selection challenge in edge computing, encompassing heuristic, optimization-based, machine learning, and reinforcement learning techniques as shown in Table 5.1. We recognise the limitations of these approaches in dynamic settings and emphasise our contributions, focusing on contextual data, immediate feedback, and expert-informed decision-making.

Table 5.1: Comparative Analysis of Node Selection Techniques in EC

Method	**Main Focus**	**Technique Used**	**Key Differentiators**
Heuristic Approaches	Static node selection	Greedy strategies, proximity-based selections	Assumes static environments, lacks adaptability
Optimization-Based Methods	Optimize capabilities and minimize latency	Mixed-integer programming, meta-heuristic algorithms like GA, PSO, ACO	NP-hard, requires relaxations, focuses on computational constraints
Machine Learning Models	Improve latency and response time	Supervised learning, data partitioning	Requires historical data, not suited for dynamic feedback
Reinforcement Learning Techniques	Minimize latency and energy	Deep reinforcement learning, gradient policies, decentralized RL	Trained on historical feedback, lacks immediate adaptability
ExpGradBand (Our Approach)	**Dynamic EC environments**	**Contextual MAB**	**Uses real-time contextual info and feedback, improves decision accuracy in dynamic environments**

We summarize the contribution of this chapter: (i) **Utilizing Contextual Information:** Our approach integrates contextual information from current and past nodes, facilitating better-informed decision-making. This differs from previous MAB algorithms that often exclude past and contextual information, therefore enhancing the precision of node selection. (ii) **Immediate Feedback Integration via Expert Pool:** We provide a feedback-driven method that uses a group of experts to analyze instant feedback. This expert-driven technique guarantees fast adaptability to evolving circumstances in the EC environment, making our solution more re-

sponsive than traditional MAB tactics. (iii) **Comparative Performance Analysis:** We provide comprehensive comparison assessments of our methodology versus current MAB techniques, demonstrating that our solution attains enhanced performance in dynamic EC situations. This demonstrates the efficacy and usefulness of our methodology in real-world situations.

Notation	Definition
$\mathcal{N}$	Set of available nodes.
T	Finite time horizon.
t	Time instance, $t \in \{1,2,\ldots,T\}$.
I_t	Node selected at time t.
$r_{i,t}$	feedback of node $n_i \in \mathcal{N}$ at time t.
U_t	Cumulative reward up to time t.
p_i	Selection probability of node n_i.
h_i	Preference score of node n_i.
η	Learning rate parameter.
$w_{i,t}$	Weight assigned to node n_i at time t.
γ	Exploration–exploitation parameter.
z_t	System state at time t.
ϕ_1, ϕ_2	AR(2) coefficients for system state evolution.
ε_t	Noise term in the AR model.
$R(T)$	Per-round regret after T rounds.
s_t	Indicator variable: 1 if optimal node selected at round t, else 0.

Table 5.2: Notations of parameters

5.3 Background & Preliminaries

In this section, we present the essential concepts and formal definitions necessary for comprehending the node selection problem in edge computing. We present the multi-armed bandit (MAB) framework as a reinforcement learning methodology to conceptualize the problem and delineate essential components, including the system model, rewards, and feedback mechanisms. This context establishes the foundation for the suggested approach, which utilises contextual feedback and adaptive modifications in adversarial settings.

The MAB problem is a single-state reinforcement learning problem where a fixed limited set of resources must be allocated among competing choices to maximize the expected gain. In our context of EC, the choices correspond to available nodes, where the node selection problem can be viewed as a bandit game between a decision-maker (node selection algorithm) and the environment. The environment captures the dynamic non-stochastic elements of the EC ecosystem. Formally, let us consider a discrete time domain $t \in \{1,2,\ldots,T\}$ with finite horizon $T > 0$. We assume an edge network of nodes represented by the graph $\mathcal{G}(\mathcal{N},\mathcal{E})$, where $\mathcal{N}$ denotes

the set of nodes (vertices), as defined in Table 5.2, and $\mathscr{E}$ denotes the set of edges. Nodes n_i and n_j directly communicate bilaterally if there exists an edge $e_{ij} = e_{ji} \in \mathscr{E}$. When a new task is received from an EC application, then the node selection mechanism starts off to seek and finally select the most suitable node to execute the task within the finite horizon T.

The mechanism decides on selecting a node from $\mathscr{N}$ at time t and receiving a *reward* (which will be elaborated later). Let us define with $r_{i,t}$ the reward of a node $n_i \in \mathscr{N}$ at time t. Let also $I_t \in \mathscr{N}$ be the node selected by the mechanism at time t. The objective of the mechanism is to sequentially probe nodes' signals from $\mathscr{N}$ at every time t such that the cumulative reward U_t up to time t is maximized over the selected nodes $\{I_1, \ldots, I_t\}$, i.e.,

$$U_t = \max \sum_{\tau=1}^{t} r_{I_\tau,\tau}. \tag{5.1}$$

We then formulate the node selection problem as follows.

Problem 3. *Consider a set of available nodes $\mathscr{N}$ and an incoming task in an EC environment. At every time t, $1 \leq t \leq T$ with finite horizon $T > 0$, find the best node $I_t \in \mathscr{N}$ such that the cumulative reward U_t defined in Equation (5.1) is maximized.*

It should be emphasized that, in our case, we make no statistical assumptions about the nature of the node selection process generating the reward feedback signals $\{r_{i,t}\}_{t=1}^{T}$ from all the nodes $n_i \in \mathscr{N}$. Rather than assuming a well-behaved stochastic process with a fixed but unknown distribution, we consider that each node n_i is assigned to an arbitrary and unknown sequence of rewards, potentially dependent on past rewards. These rewards $r_{i,t} < \infty$ are chosen from a bounded real interval, which vary with each time step t. Each time t, a node I_t is selected. Then, the mechanism receives the corresponding reward $r_{I_t,t}$ from the sequence assigned to that node.

In standard MAB settings like online news recommendation and advertising, feedback reward signals like user clicks can be easily obtained. The rewards serve as a guide in the selection mechanism, enabling it to continuously improve its performance by prioritizing actions that *regularly* provide higher rewards. Typically, rewards are assumed to be the binary feedback signals, e.g., $r_t \in \{0,1\}$ reflecting reactions obtained from the environment for failure or success, respectively, given a chosen action. In our node selection problem, binary feedback is not practical. Instead, we should use continuous feedback values in the rewards. In general, feedback can be e.g., heart-beat signals obtained from the nodes that can give the current state of the node in the network. For instance, the reward can reflect the percentage of the offloaded tasks a node is currently performing, or the percentage of CPU load, or the ratio of response time out of the required time limit [109, 110].

In this work, we assume the feedback reward model using heart-beat signals as in [124]. Our proposed method assumes that the nodes follow standard distributed system protocols and communicate using heart-beat signals [109]. This heartbeat signal encodes the system state as represented using performance measures load, number of active tasks, memory usage, etc. Deviating from standard MAB formulations with binary feedback, our feedback rewards are positive real values $r_t > 0$. Moreover, as mentioned in Section 5.1, the system state evolves sequentially which at any point in time might depend on the system state at immediate past time. Such a sequential nature stipulates auto-regressive models to model feedback rewards [7, 125].

5.4 Bandit Mechanisms for Edge Node Selection

This section offers adaptations of multi-armed bandit (MAB) algorithms designed for node selection challenges in edge computing contexts with non-stochastic feedback rewards, where reward signals obtained from node observations (e.g., latency, load, or connectivity quality) are not assumed to follow a stationary probabilistic distribution but instead depend on the current system state and recent operational conditions. We concentrate on non-stochastic bandit mechanisms, encompassing adaptations of Thompson Sampling (TS), Gradient Bandits (GB), and the Exponential-weight for Exploration and Exploitation (Exp3) method. Each method is examined with its formulation, algorithmic specifics, and significance to the dynamic, feedback-oriented characteristics of the challenge at hand.

We firstly propose extensions of standard MAB algorithms for our Problem 3 under non-stochastic feedback rewards. As we focus on non-stochastic MABs, we exclude the extensions of the popular stochastic algorithms like UCB and ε-greedy proposed for node selection in EC [98]. Instead, we focus on probabilistic bandit algorithms in non-stochastic settings.

5.4.1 Node Selection with Thompson Sampling under non-binary Rewards

Thompson Sampling (TS) is a lightweight and relatively simple MAB algorithm based on the Bayesian principle. TS manages the exploration-exploitation trade-off by maintaining a probability distribution for each node's potential reward and then sampling from these distributions to steer the decision-making. The prior distribution is continuously updated based on observed rewards.

We leverage the randomized selection strategy used in TS adversarial environments [126]. Specifically, in our case, given that the rewards are positive real numbers, we follow the principles of extending TS to accommodate non-negative rewards [7]. Therefore, we model the rewards for each node using a log-normal distribution, since rewards are strictly positive and their logarithmic transformation enables a Gaussian likelihood. This, in turn, allows the use

of Gaussian-Gamma conjugate priors for tractable Bayesian updates of the mean and precision parameters. For the conjugate priors of the *mean* and the *precision* (inverse of the variance) of the underlying normal distribution, we define the following modeling distributions for the reward:

$$\tau_{i,t+1}|r_{i,t} \sim \Gamma\left(\alpha + \frac{\nu_i}{2}, \beta + \frac{1}{2}\text{SSE} + \frac{\nu_i}{2\,(\nu_i+1)}\,(\bar{u}_{i,t} - u_0)^2\right) \tag{5.2}$$

$$\mu_{i,t+1}|r_{i,t}, \tau_{i,t+1} \sim \mathcal{N}\left(\frac{\nu_i \bar{u}_{i,t} + 1}{\nu_i + 1}, \frac{1}{(\nu_i+1)\tau_{i,t+1}}\right), \tag{5.3}$$

where $\tau_{i,t}$ and $\mu_{i,t}$ represent the estimated precision and mean sampled from `Gamma` and `Gaussian` distributions after the observation of the rewards for the node n_i at time t, respectively. Moreover, ν_i represents the number of times the node n_i is selected by the mechanism up to time t, while α, β, and u_0 are fixed known constants (elaborated later). In addition, the 'logarithmized' reward $u_{i,t} = \ln(r_{i,t})$ derives from the reward $r_{i,t}$ associated with node n_i at time t. The term $\bar{u}_{i,t}$ corresponds then to the empirical mean of logarithmized rewards $\{u_{i,1}, \ldots, u_{i,t}\}$ calculated up to time t. Hence, the sum of squared errors SSE in (5.2) is defined as $\text{SSE} = \sum_{k=1}^{t}(u_{i,k} - \bar{u}_{i,k})^2$.

Given the prior mean vector $\mu_t = [\mu_{1,t}, \ldots, \mu_{K,t}]^\top$ and precision vectors $\tau_t = [\tau_{1,t}, \ldots, \tau_{K,t}]^\top$ for all the nodes $n_i \in \mathcal{N}$, all the rewards $\mathbf{r}_t = [r_{1,t}, \ldots, r_{K,t}]^\top$ in TS at time t are sampled according to:

$$\mathbf{r}_t|\mu_{t-1}, \tau_{t-1} \quad \sim \text{LogNormal}\left(\mu_{t-1}, \sqrt{\tfrac{1}{\tau_{t-1}}}\right). \tag{5.4}$$

The TS mechanism for node selection Problem 3 is provided in Algorithm 6. At every time t, the mechanism chooses the node I_t with the highest posterior reward value for potentially executing the required task or service. The mechanism then retrieves the reward $r_{I_t,t}$ of the chosen node I_t and updates the corresponding mean and precision values of its reward distribution accordingly. At the end of horizon T, the mechanism returns the best node maximizing (5.1). The values of the constants (α, β, u_0) are set to $(K, 1e-3, 0.05)$, respectively based on [125]. To encourage initial exploration by optimistic initialization, the initial values of the mean are set to $\log 2$ and for precision to $\frac{1}{0.05}$ as suggested in [112, 125].

Algorithm 6: TS Mechanism for Node Selection

Parameters: Number of nodes K, horizon T.
Initialize : $\mu_{i,1} = \log 2, \tau_{i,1} = \frac{1}{0.05}, \forall\, i = 1, \ldots, K$.
for $t = 1, 2, \ldots, T$ **do**
  sample node $I_t = \arg\max_i \text{LogNormal}\left(\mu_{i,t-1}, \sqrt{\frac{1}{\tau_{i,t-1}}}\right)$
  select node I_t and observe node's reward $r_{I_t,t}$
  update node's $\tau_{I_t,t+1}$ and $\mu_{I_t,t+1}$ w.r.t. (5.2) and (5.3);

5.4.2 Node Selection with Gradient Bandits under non-binary Rewards

Gradient Bandit (GB) estimates the preference score of each node, which will influence the selection, using feedback rewards [112]. Given the fact that the observed rewards of the currently selected nodes vary, then evidently, the corresponding preference score also varies. A GB algorithm learns to prioritize nodes with increasing rewards over those with lower ones. This prioritization is achieved by incrementally updating specific rules based on the observed rewards and the present probability of choosing the selected node. The GB's principle is that nodes with higher preference scores have a higher probability of selection.

Recent studies have shown that GB algorithms are robust to noises while being able to find a global optimal dynamic allocation strategy faster with constant learning rates [127]. Furthermore, GB algorithms inherently guarantee "weak exploration" in which the chance of selecting each node does not degrade too rapidly. This condition guarantees that all nodes are sampled infinitely often with probability 1 and, in situations where choices have to be made in a sequential fashion, as in our context, GB algorithms are significantly efficient [128].

In our adoption of the GB sequential learning paradigm as a node selection mechanism, we use the widely applied softmax function to 'transform' the preference scores into probabilities. This guarantees that nodes with higher levels of preference have a greater chance of being selected while permitting the exploration of less preferred nodes. We introduce the GB mechanism in Algorithm 7. As in TS, the GB mechanism input is the number of nodes K and time horizon T. The mechanism starts by initialising the *preference* vector $\boldsymbol{h} = [h_1, \ldots, h_K]$ and empirical reward mean vector $\bar{u} = [\bar{u}_1, \ldots, \bar{u}_K]$ to zero. At every time t, the mechanism estimates the probability distribution $\mathbf{p} = [p_1, \ldots, p_K]$ over the nodes using the preference score, which is transformed to a probability (using softmax function over $\boldsymbol{h}$). The mechanism randomly selects a node I_t following the estimated probability distribution $\mathbf{p}$ at time t. Then, the mechanism incrementally updates the corresponding mean reward of the selected node along with the preference values of the selected and non-selected nodes at time t (lines 5-13) having learning rate $\eta \in (0,1)$.

5.4.3 Node Selection with Exponential-weight for Exploration & Exploitation

The MAB algorithm Exponential-weight for Exploration & Exploitation (Exp3) is proposed to deal with scenarios in which the distribution of rewards is not fixed and might vary over time [113, 114]. In our context, the node selection mechanism based on Exp3 keeps track of the probability distribution of all nodes. This distribution is used for choosing the most appropriate node at time t. The likelihood of choosing each node is adjusted according to the rewards obtained from the selected nodes up to t. Nodes with previous relatively high rewards are assigned to higher weights influencing future selection decisions. The weight of each node

Algorithm 7: GB Mechanism for Node Selection

Parameters: Number of nodes K, horizon T
Initialize : $\boldsymbol{h}_1 = \mathbf{0}, \bar{u}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  $p_{i,t} = \frac{\exp h_{i,t}}{\sum_\ell \exp h_{\ell,t}}, \forall\, i = 1 \ldots K$
  sample node $I_t \sim \text{sample}(\boldsymbol{p}_t)$
  select node I_t and observe node's reward $r_{I_t,t}$
  $\nu_{I_t} \leftarrow \nu_{I_t} + 1$
  update mean reward $\bar{u}_{I_t} = \bar{u}_{I_t} + \frac{1}{\nu_{I_t}}(r_{I_t} - \bar{u}_{I_t})$
  for *node* $n_i \in \mathcal{N}$ **do**
    if $n_i = I_t$ **then**
      node n_i is selected
      update preference $h_{i,t} = h_{i,t} + \eta(r_{I_t} - \bar{u}_{I_t})(1 - p_{i,t})$
    else
      node n_i is not selected
      update preference $h_{i,t} = h_{i,t} - \eta(r_{I_t} - \bar{u}_{I_t})p_{i,t}$

$w_{i,t}$ is updated depending on the observed reward $r_{i,t}$ in an exponential manner with exponent $\gamma \in (0,1)$. The exponential weighting guarantees that it places a greater priority on nodes that have been successful (w.r.t. high reward values) while still preserving a certain degree of exploration in order to identify potential nodes in the future that may be of greater potential [129].

The Exp3 mechanism for node selection is provided in Algorithm 8. At every time t, the selection probability distribution $\mathbf{p}_t = [p_{1,t}, \ldots, p_{K,t}]$ is updated based on nodes' weights. The weight of the selected node I_t depends on the current reward $r_{I_t,t}$ and the associated probability of selection $p_{I_t,t}$. For the rest of the nodes, which have not been updated, their weights are not updated at time t.

5.5 Node Selection based on Contextual Data

In practice, one could make use of the contextual data (side information) to improve the efficiency of the bandit algorithm. Context data can improve the bandit algorithms as it provides an easy way to obtain full information, with uncertainty, regarding the actions [101, 130]. Depending on the application settings, one could make use of different kinds of contextual data. In the case of online advertisement, this can be the information associated with the customers and ads. In the edge computing model, since the nodes only communicate with heartbeat signals, we can consider the past heartbeat signals as the context data. The idea is to make use of the past data to train a supervised oracle to predict the future values of the nodes. In the past expert based MAB strategies are proposed that make use of historical data to train supervised oracles that the decision-maker can rely on [113, 131]. So far, expert based algorithms are not tested

Algorithm 8: Exp3 Mechanism for Node Selection

Parameters: Number of nodes K, horizon T, exponent γ
Initialize : K-dimensional weight vector $\boldsymbol{w}_1 = \mathbf{1}$
for $t = 1, 2, \ldots, T$ **do**
  selection probability $p_{i,t} = (1-\gamma)\frac{w_{i,t}}{\sum_\ell w_{\ell,t}} + \frac{\gamma}{K}, \forall i = 1, \ldots, K$
  sample node $I_t \sim \text{sample}(\boldsymbol{p}_t)$
  select node I_t and observe node's reward $r_{I_t,t}$
  for *node* $n_i \in \mathcal{N}$ **do**
    if $n_i = I_t$ **then**
      node n_i is selected
      $\hat{r}_{i,t} = \frac{r_{i,t}}{p_{i,t}}$
    else
      node n_i is not selected
      $\hat{r}_{i,t} = 0$
    exponential weight update $w_{i,t+1} = w_{i,t} \exp\left(\frac{\gamma \hat{r}_{i,t}}{K}\right)$

for node selection in edge computing. As expert-based algorithms have access to both current and past values, one would expect it to perform better compared to non-expert-based MAB algorithms. In this section, we propose an extension to the popular contextual algorithm, namely Exponential-weight algorithm for Exploration and Exploitation using Expert advice (Exp4), for node selection and propose a variant of the Gradient Bandits that makes use of contextual data.

5.5.1 Exponential-weight algorithm for Exploration and Exploitation using Expert advice (Exp4) for Node Selection

The pseudo-code for the Exp4 algorithm for node selection is given in Algorithm 9. The Exp4 algorithm works by combining advice from multiple experts. The algorithm starts by assigning equal weights to all experts, assuming they are equally trustworthy. In each round, the experts provide advice in the form of probability distributions over the nodes. The algorithm combines this advice into a weighted probability distribution and chooses the node based on this combined advice. The algorithm updates the weights of the experts after observing the rewards. Experts who gave good advice (leading to high rewards) get their weights increased, while those who suggested poor actions have their weights reduced [113].

Algorithm 9: Exp4 Algorithm for Node Selection

Parameters: Number of experts N, Number of nodes K, horizon T, exploration parameter $\gamma \in (0,1]$

Initialize : $w_{i,1} = 1$ for $e_i = 1, \ldots, N$

for $t = 1, 2, \ldots, T$ **do**

  Get expert's advice vectors $\{\xi_t^1, \ldots, \xi_t^N\}$, each vector ξ_i is a distribution over K nodes

  for *node* $n_j \in \mathcal{N}$ **do**

    $p_{t,j} = (1-\gamma)\sum_{i=1}^{N} \frac{w_{i,t}\xi_{t,j}^i}{\sum_{i=1}^{N} w_{i,t}} + \frac{\gamma}{K}$

  sample node I_t according to $\mathbf{p}_t$, and receive reward $r_{I_t,t}$

  for *node* $n_j \in \mathcal{N}$ **do**

    Calculate unbiased estimator of reward, $\hat{r}_{t,j} = \frac{r_{I_t,t}\mathbf{1}(n_j = I_t)}{p_{t,j}}$

  for *expert* $e_i \in \mathcal{K}$ **do**

    Calculate estimated expected reward $y_{t,i} = \xi_t^i \cdot \hat{r}_t$

    Update expert's weight $w_{i,t+1} = w_{i,t} \exp\left(\frac{\gamma y_{t,i}}{K}\right)$

5.5.2 Experts with Gradient Bandit

This section examines how contextual data, including historical heartbeat signals, can improve the efficacy of bandit algorithms in edge computing. We suggest enhancements to current contextual algorithms, including the Exponential-weight method for Exploration and Exploitation utilising Expert advice (Exp4), and build on the variant of Gradient Bandit that integrates contextual information introduced in Chapter 4. These strategies utilise historical data and expert predictions to enhance decision-making and adaptation in dynamic contexts.

We build on the extension to the Gradient Bandits algorithm that incorporates contextual data to enhance decision-making. Our approach revolves around decoupling the accuracy of expert predictions from the performance of the nodes. To achieve this, we maintain two distinct distributions: one over the experts and another over their predictions. We utilize a preference-based probability distribution estimation technique for choosing the nodes, similar to the original GradBand method. Meanwhile, the experts are weighted based on their prediction accuracy, allowing us to give more importance to experts who have demonstrated better performance. As in the Exp4 algorithm, we train supervised oracles to predict outcomes. However, instead of selecting a node at random, we first sample an expert and then apply the GradBand algorithm based on that expert's advice. We refer to this method as **Exp**ert based **Grad**ient **Band**it (Exp-GradBand), following its original introduction in Chapter 4. The pseudo-code of the algorithm is provided earlier in Algorithm 5.

Table 5.3: Hyperparameters of the edge nodes

#	ϕ_1	ϕ_2	**mean**
1	0.25	-0.25	$\exp(1.50)$
2	-0.45	0.25	$\exp(0.50)$
3	-0.35	0.40	$\exp(0.55)$
4	0.25	-0.30	$\exp(0.75)$
5	-0.25	0.30	$\exp(0.25)$

5.5.3 Experimental Results

In this section, we compare the performance of different algorithms for node selection problem.

To carry out experiments, we generate synthetic non-stochastic data. As we argued in the introduction 5.1, a system state can be seen as a time series that depends on the immediate past states. Hence we use a simple Autoregressive (AR) process to model the system state. An Autoregressive (AR) model is a statistical model used for analyzing and forecasting time series data. The core idea behind the autoregressive model is that the current value of a time series depends linearly on its previous values (lags). This model assumes that past values of the series hold important information about future behavior, so by looking at past observations, we can predict future points. An AR model is mathematically represented as $AR(q)$, where q is the number of lagged observations included in the model. Formally:

$$z_t = c + \phi_1 z_{t-1} + \phi_2 z_{t-2} + \cdots + \phi_q z_{t-q} + \varepsilon_t, \tag{5.5}$$

where z_t is the current value of the time series and $z_{t-1}, z_{t-2}, \ldots z_{t-q}$ are the q previous values (lags). The constant term c is the intercept, and $\phi_1, \phi_2, \ldots, \phi_q$ are the coefficients (weights) that describe how much influence each lagged value has on the current value. ε_t represents the error term, which accounts for randomness or noise that is not explained by the past values. The AR model is considered a linear regression model but applied to time series data, where the predictors are the past values of the series itself. It is especially useful when the data shows dependencies over time but does not have strong seasonal patterns or trends that require more complex modeling. We assume that the future system state is linearly dependent on the two immediately preceding states and thus use a simple AR(2) model.

We simulate a controlled setting involving five edge nodes and ten connected devices. This experimental scale is appropriate for the purpose of this study, as it allows a clear and tractable evaluation of the node selection mechanism and its response to dynamic heart-beat feedback. At every time interval, the nodes send heart-beat signals that encode their current state. We assume that these heart-beat signals encode the system state in such a way that higher values correspond to more suitable nodes for selection. The different parameters for the nodes are given in Table 5.3. Evaluating the effect of larger network scales, with more nodes and connected devices, remains an important direction for future work.

In our experiments with contextual bandits, we use deep learning-based time series predictors as experts. We used three different deep-learning models: *(i)* Feedforward Neural Network *(ii)* Recurrent Neural Network and *(iii)* Long Short- Term Memory Network.

Feed Forward Network A Feedforward Neural Network (FNN) is the simplest form of neural network used for time series prediction. It is a uni-direction network where the data is passed from input nodes, through hidden layers, to output nodes—without looping back. FNN can be used as an autoregressive model by predicting the next value in a sequence based on previous inputs. In this setup, the FNN takes a fixed-length window of past values from a time series as inputs and outputs the next predicted value in the sequence. In our experiments, we set the window size to 16 and two hidden layers.

Recurrent Neural Network Recurrent Neural Networks (RNNs) are designed to model sequential data by capturing temporal dependencies. Unlike feedforward networks, RNNs have connections that allow information to persist by using loops, enabling them to maintain a hidden state that can process inputs over time. This makes RNNs particularly well-suited for tasks involving time series, where the order and context of input data are important. In our experiments, we used the same model configuration as in the case of FNN.

Long Short-Term Memory Long Short-Term Memory (LSTM) is a type of RNN designed to capture long-term dependencies in sequential data by using memory cells to retain important information over time. We used LSTM as an autoregressive model to predict the next value in the time series by learning patterns from previous data points. In this setup, the LSTM processes a series step-by-step, using its memory cells to track and store relevant past information while forgetting irrelevant details. At each time step, it predicts the next value in the sequence, and this prediction can be fed back into the model as input for future predictions. This autoregressive approach allows LSTMs to be highly effective in time series forecasting, speech generation, and text generation, where capturing both short- and long-term dependencies is crucial. In our experiments, we used the same model configuration as in the case of FNN.

5.5.4 Protocol & Metrics

We generate a data stream corresponding to two hours of operation with heartbeat signals sent every second. We use the first hour of data to train experts, and the last hour of data is used to compare the performance of all algorithms. The performance of different algorithms is evaluated using the per-round regret. Formally, regret is defined as the loss suffered by the algorithm relative to the optimal strategy. After T rounds, the per-round regret of an algorithm is defined as:

$$R(T) = \frac{T - \sum_{t=1}^{T} s_t}{T}. \tag{5.6}$$

where s_t is the binary variable which is equal to one if the optimal node is selected at round t and zero otherwise. The hyper-parameters used in the experiments for all algorithms are given in Table 5.4.

Table 5.4: Hyperparameters of different algorithms

GradBand	Exp3	Exp4	ExpGradBand
$\alpha = 0.01$	$\gamma = 0.01$	$\gamma = 0.01, \eta = 0.01$	$\alpha = 0.01, \eta = 0.01$

5.5.5 Experimental Evaluation under Delayed Feedback

In EC environments, the feedback might not be processed immediately due to various runtime constraints. In the delayed feedback experiments, we try to quantify the impact of this delay through simulations. The feedback is received only after a fixed time unit and this delay is applied consistently across all evaluated algorithms. The per-round regret for a delay of five time units is illustrated in Figure 5.1.

The efficiency of algorithms designed for dynamic and hostile contexts, like those in edge computing, is significantly impacted by the amount of tardy feedback. ExpGradBand is vulnerable to issues highlighted by Figures 5.1 and 5.2, which show the influence of delayed input on various algorithms. Figure 5.1 illustrates the performance of several bandit algorithms, including the newly suggested ExpGradBand, across many rounds in terms of cumulative regret per round. GradBand demonstrates a constant, gradual decrease in cumulative regret over time, while ExpGradBand, which depends on immediate feedback, has more pronounced issues. The detrimental effect of postponed feedback on ExpGradBand is evident, as it consistently underperforms GradBand even after several iterations. ExpGradBand, which relies on timely feedback for adaptive modifications, has a drawback when feedback arrives late, resulting in increased cumulative regret.

Figure 5.2 offers a complementary perspective by showing how the cumulative regret changes with different levels of delayed feedback. Unlike Figure 5.1, where the delay is fixed and the cumulative regret is observed over the rounds, here we vary the number of consecutively delayed feedback observations and report the corresponding cumulative regret. In this instance, ExpGradBand is substantially more impacted by delayed feedback compared to GradBand and Thompson Sampling. ExpGradBand's efficiency suffers from the unpredictability and latency of postponed feedback, in contrast to GradBand, which maintains a more constant and smaller cumulative regret. Both Figures 5.1 and 5.2 show how decisions are sometimes taken without immediate feedback due to these postponements. When weighed against GradBand in situations with delayed input, ExpGradBand regularly performs worse, highlighting this weakness in both its initial and long-term performance.

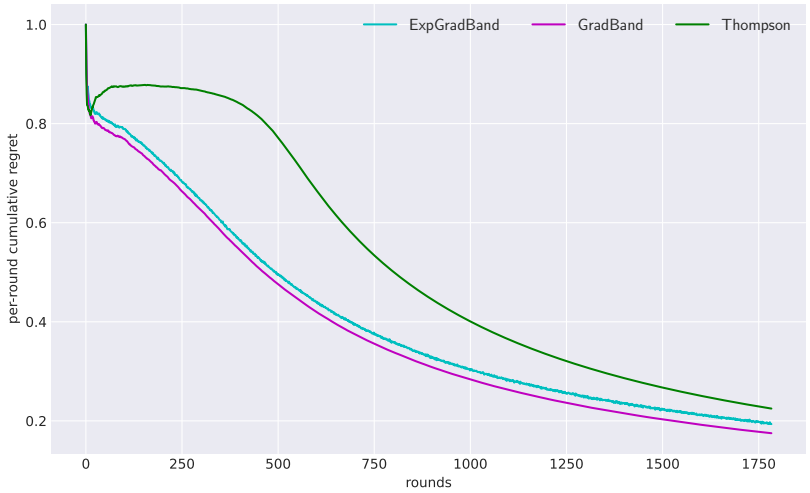


Figure 5.1: Per-round Regret when five consecutive feedback signals are delayed.

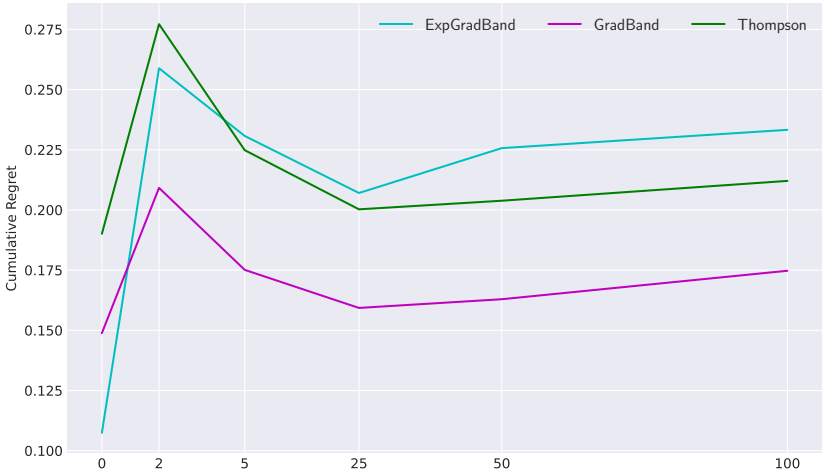


Figure 5.2: Regret as a function of delayed feedback.

5.5.6 Experimental Evaluation under Lost Feedback

In this setup, we simulate the setting where the feedback signals are lost. We plot the per-round regret of different algorithms for a setting where five consecutive feedback signals are lost after every five time intervals in Figure 5.3.

Algorithm performance is severely affected by missing feedback; feedback signals are crucial for fine-tuning methods and achieving better results over time. Lack of input makes it harder for algorithms to fine-tune their performance, which in turn leads to less-than-ideal outcomes. The offered figures aid in demonstrating the significance of missing feedback and its effects

on various algorithms, ExpGradBand included. Figure 5.3 illustrates cumulative regret among algorithms, offering a comprehensive perspective on their overall performance in the context of feedback loss. It displays the cumulative regret per round for several bandit algorithms, such as ExpGradBand, GradBand, and Thompson Sampling, over a significant number of rounds. ExpGradBand is especially vulnerable in the early phases when input is absent. In the first 700 iterations, ExpGradBand has difficulties in matching the performance of GradBand. This period of fragility and subsequent recovery demonstrates the robustness of ExpGradBand, especially its capacity to ultimately surpass competing algorithms despite initial challenges. After about 700 iterations, ExpGradBand surpasses the other algorithms, including GradBand, suggesting that it finally learns and utilizes its dynamic modification skills more efficiently. The current phase of recuperation demonstrates the capability of ExpGradBand to thrive in difficult circumstances, as long as it has the time to adjust.

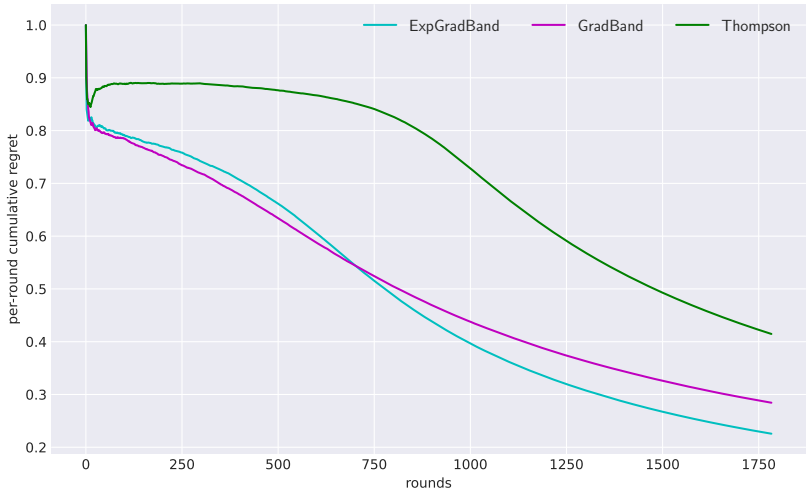


Figure 5.3: Per-round Regret when five consecutive feedback signals are lost.

Figure 5.4 offers a more in-depth examination of the initial rounds, where the effect of non-existent feedback is most evident. ExpGradBand has an excellent start, as seen by its initial performance, which is superior to that of both GradBand and Thompson Sampling from the beginning. GradBand, on the other hand, surpasses ExpGradBand as the rounds proceed. Thompson Sampling tends to exhibit the highest cumulative regret across most of the examined settings, although its performance becomes closer to the other methods at higher levels of lost feedback. Despite originally being in the lead, ExpGradBand soon encountered difficulties but was still able to perform effectively. The resilience exhibited by ExpGradBand in response to feedback variability highlights its strength and the efficacy of its adaptive mechanisms, which may be essential for implementation in settings where feedback consistency is not assured. In conclusion,

the absence of feedback is a substantial difficulty in dynamic scenarios such as edge computing, especially for algorithms that rely on real-time input. These statistics emphasize both the early-stage weaknesses of the algorithm as well as its potential for recovery and adaptability as time goes on.

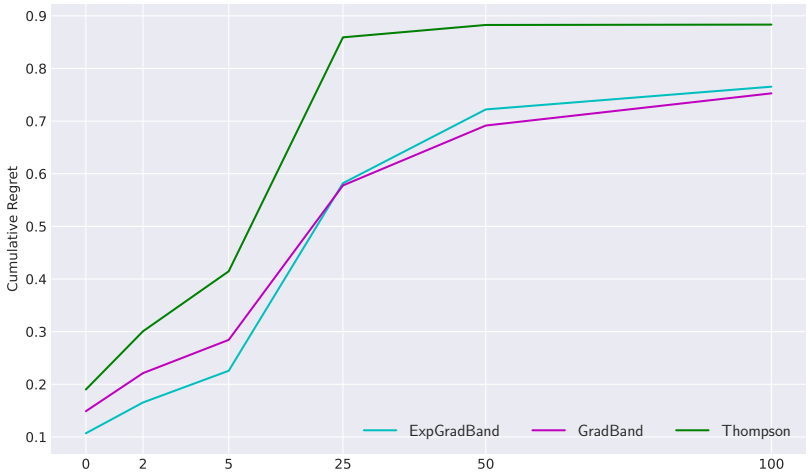


Figure 5.4: Regret as a function of lost feedback.

5.6 Conclusions

Using the Multi-Armed Bandit (MAB) online learning paradigm, we provide an enhanced decision-making approach for node selection in Edge Computing. We introduce ExpGradBand, which dynamically balances exploration and exploitation in order to optimize node selection. ExpGradBand performs much better than other MAB approaches and baseline solutions in terms of efficiency and resource utilization, as shown by the results of our experiments. In addition, the ExpGradBand model is advantageous for deployment in edge nodes since it is capable of achieving the maximum reward values while incurring the least amount of processing overhead. Our research agenda includes studies on how ExpGradBand is affected by the peculiarities of mobile applications and timeliness of data, with the end goal of developing improved node selection mechanisms in mobile computing environments that take into account these aspects.

Chapter 6

Thompson Sampling-based Recursive Block Elimination for Dynamic Assignment under Limited Budget in Pure-Exploration

6.1 Introduction

It is common for online business providers to market different variants of an entity. Here, the term 'entity' is used in a very generic sense, referring to any application-specific object such as a service, product, or even a machine learning model. Instances of such online business practices can be observed in various domains, including logistics, mobility management, e-commerce, and big-data management systems, among many others. In logistics, it is very common for users to choose from different delivery services for the same source-destination pair. For example, DHL provides express and economy services, while UPS offers standard and express services. Mobility management service providers like Bolt, Lyft, etc., allow customers to select from different variants of ride services, such as standard or shared. Similarly, in e-commerce, it is common to find different variants of the same product available for purchase, such as a pack of five or ten chocolate bars. Many standard problems that arise in data management and machine learning systems can also be interpreted through the same lens, namely, as the selection of the most suitable variant among multiple alternatives under resource and performance constraints. In Federated Learning (FL), pruning techniques are used to compress global predictive models for different compression (pruning) ratios. These compressed models are then distributed among edge nodes (e.g., end-user devices) [132] for different prediction tasks. These compressed models can be considered as variants of the global model. Another closely related problem is finding the optimal hyperparameters of a machine learning model during the training phase, as introduced in [133].

A recurring problem across these aforementioned application domains is the challenge of finding the optimal values to be assigned to different services/products to maximize expected rewards or revenues. In logistics operations, including mobility management and e-commerce, this problem is akin to determining the optimal prices for different variants of services (e.g., standard vs. express) to maximize revenue. In the case of FL pruning problem, the optimal pruning ratio varies for different nodes due to the heterogeneity of the edge nodes and network infrastructure. The objective here is to find pruning ratios that minimize communication and computational costs [132]. The primary challenge in these problem settings is that the demand curve for services and products or the data distribution at the networked nodes is not known to the decision-maker (algorithm) in advance. Nonetheless, the decision-maker has the flexibility of continuously changing the assigned values (e.g., pruning ratios or prices) and often randomizing those changes to 'learn' the optimal values. Historically, multi-armed bandit (MAB) algorithms have been employed to learn optimal assignment values adaptively. In practical settings like logistics and e-commerce, price experiments are conducted using the principle of "learn, then earn" to estimate optimal prices. In contrast, MAB algorithms provide the flexibility of "learn-while-earn" [134], which enables faster data-driven decision-making and enhances the overall experimental design. MAB-based adaptive learning algorithms have been proposed for dynamic pricing [134–137], as well as for determining optimal pruning ratios in federated learning [132].

A Multi-armed bandit (MAB) is a feedback-driven sequential learning framework that models the trade-off between *exploration* and *exploitation* over a sequence of actions [100, 101]. In the MAB framework, the decision-maker sequentially explores the set of actions to find the action with the optimal reward. In the past, MAB-based algorithms have been employed in web-based news recommendation systems [102] and online advertising [138]. In the classical stochastic reward MAB model, the focus is on the exploration-exploitation trade-off. This involves exploring the action space to estimate rewards and exploiting the most rewarding actions from the explored set to minimize cumulative regret. In this work, we consider the problem of optimal dynamic assignments in pure-exploration settings. Unlike in classical settings, in the pure-exploration setting, there is no explicit trade-off between exploration and exploitation. In the pure-exploration setting we consider, the goal is to find the best-performing action within a given fixed but **limited** budget T. That is, after exploring the actions for a relatively small number of rounds, often much smaller than the number of actions k, the decision-maker recommends an action that it considers the best one. Importantly, this recommended action need not be the same as the action played at the end of the exploration. Next, we argue that the pure-exploration setting aligns closely with the problem settings we discussed above.

It has been established that online prices do not change frequently and exhibit relatively long spells of fixed prices [139]. Moreover, the majority of online users are price-sensitive [140, 141]; consequently, evaluating revenues is costly due to the high sensitivity of users to price changes: lower prices may result in losses, while higher prices could lead to non-purchases.

The same cost considerations also arise in computational settings such as federated learning, where different pruning ratios correspond to different variants of the global model. In such scenarios, computational and communication costs directly impact the performance of downstream application systems. Consequently, evaluating the computational and communication costs for *every* possible pruning ratio combination becomes infeasible and may even lead to operational failures [132]. Therefore, to avoid such shortcomings, a strategy that explores the action space for a fixed but limited budget, while estimating optimal values such as pruning ratios or prices, is deemed appropriate. In this context, instead of framing the dynamic assignment as an exploration-exploitation trade-off, one can explore the actions exhaustively within the constraints of the given budget to adaptively learn the demand curve or data distribution and then exploit the most promising action based on the acquired knowledge. In the problems discussed above, the budget is limited, and refers to the number of users exposed to a particular price or the number of pruning ratio combinations attempted. These quantities are directly linked to T, the number of trials (rounds) in the MAB framework, as the number of users or pruning ratio combinations can be adjusted by setting T. In summary, MAB in pure-exploration provides the advantage of the "learn-while-earn" paradigm, allowing the identification of optimal pruning/price regions dynamically while respecting the budget constraints.

The three main characteristics of the problems we study here are:

- The action space is infinite, as actions correspond to assignment values spanning a given range (e.g., pruning ratios in the range (0,1))
- The reward shows structural properties, based on the observation that similar actions result in similar rewards. This is assumed in various problems, including continuous Multi-Armed Bandits (MABs) [142].
- The budget is limited, meaning the number of rounds T is much smaller than the number of candidate actions k that would arise from discretizing the continuous action space. This constraint is crucial, as the number of possible actions is prohibitively large to evaluate even a representative subset within the available budget.

It is already established that classical Multi-Armed Bandit (MAB) algorithms may perform poorly when applied in pure-exploration settings [143]. Pure-exploration algorithms have been historically employed for best-arm identification [143–146] and outlier detection problems [147–149]. However, applying such algorithms directly to dynamic assignment problems with a limited budget is challenging, as they require testing *all* action combinations *at least* once. Moreover, these algorithms assume a finite and relatively small action space [144, 146–149]. Most infinite or continuous-armed bandit algorithms leverage the structural properties of the action space (e.g., Lipschitz property) and discretize the action space to derive efficient algorithms [150, 151]. Unfortunately, these algorithms, often based on the popular Upper Con-

fidence Bound (UCB) algorithm [152], also require sampling each discretized action at least once. Given the impracticality of evaluating revenues or estimating network costs by enumerating all combinations, and considering the high cost associated with these evaluations, sampling all actions even once becomes unfeasible.

A possible workaround is to use coarse-grained discretization to limit the number of actions. However, such an approach might result in the optimal action or nearly optimal action to be excluded from the discretized action set. An orthogonal approach relies on *action elimination*, where potentially sub-optimal actions are removed during exploration based on an estimated confidence value [146, 153, 154]. To the best of our knowledge, all the elimination-based Multi-Armed Bandit (MAB) algorithms proposed in the literature so far either require sampling each action at least once [153, 154] or involve sampling only a representative action, such as the most densely connected one [146]. In this chapter, we contribute a novel *block elimination* MAB algorithm. Our approach is distinct from previous elimination-based pure-exploration algorithms, as the decision-maker recursively eliminates *blocks of actions* by estimating a confidence score over the blocks. Our algorithms are based on Thompson Sampling, a simple yet competitive heuristic for Multi-Armed bandit problems with a provable optimality guarantee [155–157]. Traditionally, Thompson Sampling has been used in classical settings, with the top-two sampling version of Thompson Sampling often employed in pure-exploration settings [158, 159]. In addition to the vanilla Thompson Sampling-based schemes, we also propose top-two variants of the block elimination strategy we develop. Unlike previous approaches, our mechanism does not necessitate sampling all actions before deciding whether an action should be eliminated or not. This enhances the feasibility and applicability of our algorithm in dynamic assignment problems with limited budgets. Additionally, all the pure-exploration Thompson Sampling based schemes are designed for fixed-confidence pure-exploration problem (ref Section 6.2) [158–160], and their empirical analysis has been limited to binary rewards. We extend Thompson Sampling for fixed-budget pure-exploration with non-binary rewards.

The remainder of this chapter is organized as follows: Section 6.2 provides an overview of related work and discusses our contributions. This section covers past work on infinite or continuous Multi-Armed Bandits (MABs), structured bandits, pure-exploration MAB algorithms, and MAB algorithms for finding optimal pruning ratios and dynamic pricing. Section 6.3 elaborates on the proposed recursive block elimination algorithm, discussing different confidence score estimation techniques. We also introduce the top-two variants of our proposed algorithm. Section 6.4 reports on our experimental results based on two different dynamic assignment problems: dynamic pricing in logistics and dynamic pruning in FL. We also conduct a sensitivity analysis of the proposed algorithms. Finally, Section 6.5 concludes this chapter.

6.2 Related Work

In this section, we discuss studies related to different aspects of the problem and the application settings we consider in this chapter.

Structured MAB with Infinite Arms Many algorithms for continuous-armed (infinite arms) bandits have been proposed in the past. Multi-Armed Bandit (MAB) problems become challenging when the arm set is very large or infinite. However, in such scenarios, the expected rewards often exhibit structural properties that we can exploit when designing efficient algorithms. Many such algorithms assume that the reward function is Lipschitz continuous. In [150], the authors proposed a Lipschitz bandit algorithm that discretizes the continuous space and uses a UCB-based algorithm. However, this approach is not feasible in our case, as discretization may still result in many arms compared to the budget. Another variant of the proposed algorithm makes use of adaptive discretization, gradually zooming in on the more promising regions of the action space. Unfortunately, that algorithm also requires sampling every action at least once. In [161], continuous-armed bandits in a one-dimensional metric space were considered, whereas [162] considered continuous-armed bandits in generic metric spaces, assuming that the mean-payoff function is a Lipschitz function with respect to some dissimilarity measure defined in the space. The proposed "zooming" algorithm achieves essentially the best possible regret bound in a minimax sense with respect to the environments studied. Inspired by the zooming algorithm, [151] adopted it to non-stochastic bandit settings where the rewards are assumed to be generated by an oblivious adversarial environment. Algorithms based on tree partitions were also proposed for problems with infinite arms. [142] proposed a tree-based optimization algorithm that recursively partitions the space. While this work is similar to ours, there are significant differences. The above works do not clearly mention how the action space is discretized to sample the actions. Moreover, these works consider bandits in the exploration-exploitation setting rather than the pure-exploration setting. Most crucially, these algorithms do not eliminate suboptimal blocks from future exploration.

Pure-Exploration Algorithms In the last decade, numerous algorithms for pure-exploration bandits have been proposed. The majority of these algorithms are based on the popular UCB algorithm [152]. The pure-exploration bandit problem is studied from two standpoints: *(i)* fixed-confidence settings and *(ii)* fixed-budget settings.

In the fixed-confidence setting, the objective is to minimize the number of rounds to find the action with the given probability confidence with respect to the best action. In the fixed-budget setting, the goal is to find the best action using, at most, the given number of rounds. Fixed-confidence algorithms optimize the budget for a given confidence level, whereas fixed-budget algorithms focus on minimizing the number of trials needed to achieve a fixed confidence level. The general consensus is that the fixed-budget setting problem is significantly harder [163].

Since our work is in the fixed budget setting, we will discuss literature pertaining to fixed-budget pure-exploration bandit algorithms only. Interested readers can refer to [144, 145, 147, 154, 163] and the references therein for a detailed overview of fixed-confidence pure-exploration algorithms.

It has already been shown that elimination-based algorithms are better suited in fixed-budget settings as they can find the optimal actions provably [163]. The crux of any elimination-based algorithm is to iteratively eliminate suboptimal actions from the action set until a unique action or near-optimal action is left.

In the classical paper [144], the authors proposed the successive reject algorithm. The algorithm divides the trial budget into $k-1$ elimination phases and successively removes the arm with the lowest empirical mean at the end of each phase. One of the main drawbacks of this algorithm is that it requires every action to be sampled at least once and thus cannot be used in a situation where the action set is very large compared to the budget. Another main drawback is that the successive rejection algorithm requires prior knowledge of a problem-dependent parameter.

Sequential Halving (SH) [153] algorithm is similar to the successive reject algorithm. Like in the successive reject case, in SH, the given budget is split evenly across $\log_2 T$ elimination phases where T is the budget. Within each round, actions are sampled uniformly to update the empirical mean values. At the end of the elimination round, the worst-performing half of the actions, in terms of empirical mean, are eliminated. Like in the successive reject algorithm, SH requires every action to be sampled at least once and thus cannot be used when the action set is larger than the budget.

In [146], authors propose a block elimination-based algorithm named Sequential Block Elimination (SBE). The algorithm also splits the budget evenly across the elimination phase, and in each elimination phase, instead of a single action, a block of action is removed. The algorithm is based on using the side information of the actions. The side information is utilized by forming action blocks and sampling the central action in each round. This is based on the assumption that the central arm is the most connected action in each block. At the end of each elimination phase, the worst-performing block, in terms of the empirical mean reward of the central action, is removed. The important thing to keep in mind is that this algorithm uses block elimination, but it plays only the central action from each block and makes the elimination decision based on the mean reward of this action.

Recently, the authors in [164] proposed the variance-based rejects (VBR) algorithm. The algorithm is similar to the successive reject and sequential halving algorithm. The algorithm proceeds in phases, and in each phase, all the non-eliminated actions are sampled for a fixed number of rounds to estimate the corresponding empirical mean reward and standard deviation. The main difference between VBR and other algorithms is that the elimination criterion is based

on a confidence score on the empirical mean rewards, not solely on the empirical mean reward. All the actions whose upper bound is lower than the current maximal lower bound are eliminated after each elimination phase. This algorithm also cannot be scaled into the problem settings where the action set is much larger than the budget.

Bandits for Dynamic Assignment Problems The majority of the previous work in the dynamic assignment is restricted to the dynamic pricing problem due to its application in e-commerce. In dynamic pricing, a company is trying to automatically optimize the prices of some services/products to maximize revenue. The user arrives sequentially, and the decision-maker sets the price. The user will only purchase the product if the price is lower than their intrinsic valuation. The decision-maker will never observe the true valuation of the product, but only the revenue from the set price.

Thompson sampling is a simple yet efficient heuristic based on randomly sampling actions according to its success probability of being optimal. Recent studies showed that Thompson sampling can significantly outperform UCB-style algorithms in the classical MAB settings [155, 157]. In the past, top-two variants of the Thompson sampling algorithm were proposed for dynamic allocation in pure-exploration settings [158, 159]. These variants will also be considered later in this chapter. These algorithms are not elimination based, and at each round, the algorithm has to choose an action from the entire action set, irrespective of the suboptimality of many actions.

In [134], authors proposed a UCB-inspired dynamic pricing algorithm. Like in the UCB algorithm, the proposed algorithm starts by sampling all the price points to construct partial demand information for different prices. This demand is used to select the next price point to sample. Since this algorithm is a UCB-style algorithm, it can not be used in our settings due to the reasons outlined earlier. [135] proposed a linear contextual bandit algorithm. The work considers dynamic pricing when the number of products is too large to be handled by standard bandit algorithms. They consider slightly different settings where the product demands are assumed to follow a linear model where the time-varying dynamics of the price-demand relation are captured using a positive semi-definite matrix. The setting is very different from what we consider in this chapter. The algorithm is specifically designed for the exploration-exploitation trade-off with the assumption that the underlying model evolves over time. This algorithm also requires all price points to be sampled at least once. Recently, many contextual dynamic pricing algorithms have been proposed. [165] proposed an extension to the famous EXP4 algorithm [113]. In [166], authors extended the UCB algorithm to consider context information. The algorithm assumes the user's valuation to be a linear function of the contexts and carries out

exploration to estimate the parameters to employ UCB-style price recommendations. Both these approaches make use of context information and are designed for the case when the action set is smaller compared to the budget constraint. In this work, we consider the settings when the budget is very limited, and the number of actions is large.

Contributions: Our main technical contributions are:

- We propose a recursive block elimination method that utilizes Thompson Sampling to address dynamic assignment problems. This approach incorporates a confidence-score-based elimination strategy to improve the decision-making accuracy. Furthermore, we explore an extension of this method by integrating a top-two sampling strategy. This extension aims to further improve efficiency by leveraging additional probabilistic insights.

- Our proposed method extends Thompson Sampling to fixed-budget pure-exploration, in contrast to the more common fixed-confidence pure-exploration framework (ref Section 6.2). Additionally, departing from conventional binary reward settings used in Thompson Sampling, we extend Thompson Sampling to non-binary reward values. This is significant because, in practical dynamic assignment problems, rewards such as revenue or computational gain are naturally real-valued rather than binary, and preserving this richer feedback enables more informative learning and finer discrimination between competing actions under limited budgets.

- We introduce two distinct techniques for estimating confidence scores, drawing inspiration from popular confidence estimation schemes widely used in MAB literature. These methods adapt classical approaches to provide a more precise measurement of uncertainty in each round of decision making, catering specifically to the dynamic and stochastic nature of the environments.

- Through comprehensive experiments conducted on two distinct problem settings (dynamic pricing and dynamic pruning), we substantiate the superior performance of our algorithm compared to state-of-the-art algorithms found in the literature.

6.3 Thompson Sampling based Recursive Block Elimination Framework

6.3.1 Preliminaries

Notations Throughout this chapter, we use bold lowercase letters to denote vectors; for example, μ represents the mean vector. The individual elements within these vectors are indicated using indexed regular font. For instance, μ_i refers to the i^{th} element of the mean vector μ. We use calligraphic letters to represent sets, such as $\mathcal{A}$, representing the set of actions. Additionally, superscripts are employed to denote time rounds, allowing for temporal tracking of values. For example μ^t and μ_i^t specifically refer to the values of μ and its i^{th} element at the t^{th} round, respectively.

Background & Definitions A stochastic Multi-Armed Bandit (MAB) problem is characterized by the number of actions k, the budget or number of rounds T, and k probability distributions $\nu_1, \ldots, \nu_k$ associated with the actions $\mathscr{A} = 1, \ldots, k$. These distributions, representing the unknown reward probabilities, are undisclosed to the decision-maker. At each round $t = 1, \ldots, T$, the decision-maker selects an action I_t from the action set $\mathscr{A}$ and receives a reward z_{I_t} drawn from the probability distribution linked to action I_t. It should be noted that the received reward is independent of past actions and observations. The consideration here is in the context of the general pure-exploration setting, as introduced in [133]. In this setting, after each round t, the decision-maker commits to an action denoted as J_t, which minimizes the expected loss (maximizes expected reward) with maximal confidence. It is essential to note that, although the decision-maker observes the reward for the action I_t played in round t, the evaluation is solely based on the recommendation J_t, distinguishing it from the classical MAB setting.

In the pure-exploration setting, the metric known as *simple regret* is employed as the loss measure. Simple regret is calculated as the difference between the mean reward of the optimal action and the mean reward of the recommended action. Let $\mu_1, \ldots, \mu_k$ represent the means of the respective distributions $\nu_1, \ldots, \nu_k$, where $i^\star = \arg\max_{i \in \mathscr{A}} \mu_i$ and $\mu^\star = \mu_{i^\star}$. The simple regret for the decision-maker at round t is defined as $r_t = \mu^\star - \mu_{J_t}$. It is important to note that the simple regret is closely related to the misidentification probability, which signifies the probability that the recommendation is not optimal, denoted as $\mathbb{P}(J_t \neq i^\star)$. An insightful observation from [144] establishes that the misidentification probability serves as an upper bound for the expected simple regret, i.e., $\mathbb{E}[r_t] \leq \mathbb{P}(J_t \neq i^\star)$. This pivotal result holds true across a spectrum of scenarios where the rewards are bounded, irrespective of their values, provided that appropriate adjustments are made for the range and scale of these rewards. Moreover, this relationship underscores the significance of the misidentification probability in understanding the performance of the decision-maker in the pure-exploration context.

In the fixed budget pure-exploration setting, the decision-maker operates within a given budget of T rounds. The primary objective is to maximize the probability of correctly identifying the optimal action within these T rounds, effectively minimizing the misidentification probability. Notably, in situations characterized by a limited budget, the number of available actions (k) is often substantially larger than the allocated budget, i.e., $k \gg T$. This discrepancy is particularly evident in our specific applications, where the number of pruning ratios and price combinations far exceeds the available budgeted rounds (T). The ultimate goal is to determine the action (such as pruning ratio or price) that maximizes the reward while adhering to the constraints of the fixed budget of T rounds. In the context of dynamic pruning, the reward is gauged in terms of computational and communication gains, while in dynamic pricing, the focus shifts to revenue optimization.

We assume smooth reward functions, specifically Lipschitz continuous functions, following the approach in [150, 151, 162]. Accordingly, our problem can be framed as a Lipschitz bandit problem within an l-dimensional metric space, where the metric is induced by the Lipschitz condition [162]. In this context, $\mathscr{X}$ represents the l-dimensional metric space of actions, characterized by an unknown reward function satisfying the Lipschitz condition. In each round of the bandit process, the decision-maker selects an action (point) denoted as $x \in \mathscr{X}$ and receives a reward from the associated reward distribution linked to x. The overarching objective is to identify the action with the highest expected reward. Importantly, in the application settings under consideration, the parameter l corresponds to the number of variants or distinct features of the products or services.

Remark 1. We acknowledge that assuming Lipschitz continuity may seem like a strong condition. However, it is important to note that, to our limited knowledge, all works on continuous-armed bandits depend on some form of smoothness and boundedness assumptions. It should also be noted that Lipschitz continuity is a weaker assumption compared to continuous differentiability or convexity. Specifically, in our cases, if the clusters are predefined, local Lipschitz continuity should be sufficient.

Thompson Sampling is a Bayesian heuristic for solving multi-armed bandit problems. At each time step, the decision-maker employs Thompson Sampling by sampling the parameters of the reward distribution from a specified prior distribution. The chosen action is then determined based on the posterior probability of being the optimal action. In the specific scenario of Bernoulli-distributed binary rewards, Thompson Sampling has demonstrated superior performance compared to UCB-based algorithms, even when there is a mismatch with the prior (it is noteworthy that the conjugate prior of the Bernoulli distribution is the Beta distribution). In our case, given that the rewards are positive real numbers, we adopt a `Normal-Gamma` prior distribution to model the mean and precision (the inverse of variance) of these rewards. Consequently, the posterior distribution of the non-negative rewards, given the observed data, follows a log-normal distribution.

At round t, precision (τ_i^t) and mean (μ_i^t) of the action i is sampled from `Gamma` and `Normal` priors, respectively, as follows:

$$\begin{aligned}\tau_i^t | z_{I_1} \dots z_{I_{t-1}} &\sim \Gamma\left(\alpha + \frac{n_i^t}{2}, \beta + \frac{1}{2}\text{SSE(i)} + \frac{n_i^t(\bar{y}_i - \xi)^2)}{2(n_i^t+1)}\right) \\ \mu_i^t | \tau_i^t, z_{I_1} \dots z_{I_{t-1}} &\sim \mathcal{N}\left(\frac{n_i^t \bar{y}_i + 1}{n_i^t + 1}, (n_i^t+1)\tau_i^t\right)\end{aligned} \tag{6.1}$$

In the expressions provided in Equation (6.1), α, β, and ξ are fixed constants. The variable n_i^t represents the number of times the action i is selected up to round t. Additionally, $y_{I_t} = \ln(z_{I_t})$, where z_{I_t} denotes the reward associated with the action played at round t, I_t. The term $\bar{y}_{I_t}$ corresponds to the empirical mean of y_{I_t} calculated over the rounds up to round t. The sum of squared errors for an action i, denoted as SSE, is defined as the summation of the squared differences between y_i and the empirical mean $\bar{y}_i$. Mathematically, this is expressed as follows:

$$\text{SSE(i)} = \sum_{a=1}^{t} (y_i - \bar{y}_i)^2 [\![I_a == i]\!]$$

Given the prior mean (μ^t) and precision (τ^t) vectors, the rewards are sampled according to:

$$\mathbf{z}'_{t+1} | \mu^t, \tau^t \sim \text{LogNormal}\left(\mu^t, \sqrt{\frac{1}{\tau^t}}\right). \tag{6.2}$$

It should be noted that we use z_i, z_i' to indicate the actual observed reward and the sampled reward from the posterior distribution for action i, respectively.

6.3.2 Block Elimination Mechanism

Our pure-exploration Multi-Armed Bandit (MAB) algorithm is based on the sequential elimination of actions. The algorithm begins by discretizing the continuous action space, a common practice in Lipschitz bandits to apply stochastic MAB algorithms like UCB [150, 151]. Previous methods for finding optimal pruning ratios [132] and dynamic pricing [134] have also adopted discretized action sets. The granularity of the discretization is a key factor that controls the number of final actions to be sampled. Since our method does not require all actions to be sampled, we propose a fine-grained discretization. In our cases, we discretize all different ratio/price values independently into equal numbers so that they can be clustered (creating blocks of actions) in subsequent steps. As in other elimination-based approaches [146, 153, 164], our algorithm runs in phases. The pseudo-code for our proposed block-elimination approach is given in Algorithm 10. The algorithm takes as input the action space ($\mathcal{X}$), budget (T), the number of rounds in each phase (κ) and the number of clusters c. In each phase, we sample the actions for

κ rounds to collect the rewards associated with those actions. After the end of every phase (κ rounds), the algorithm calculates the confidence scores and eliminates action blocks based on these scores. The exact details of how block confidence scores and rewards for the non-played actions are calculated are given in the following sections.

In parallel with block elimination, our method introduces a 'zooming' mechanism that recursively clusters actions, thereby decreasing the size of blocks (i.e., the number of actions in each block) after each phase. The zooming is achieved by further clustering all remaining blocks into c clusters after the elimination round. This approach streamlines the exploration to focus on the most promising actions among the non-eliminated actions. Finally, when the number of actions becomes too few to cluster, the algorithm shifts to eliminating individual actions based on confidence scores. This process continues until the budget is exhausted or only one action remains. After each round, our algorithm returns the action with the highest mean as the recommended choice.

The means and precisions of the rewards for all discretized actions are initialized using the optimistic initialization technique (ref Section: 6.3.2.1) [167]. In line 8, the decision-maker selects the action with the highest sampled reward based on the current prior estimations of the mean and precision of the rewards. Afterward, the actual reward from the chosen action is observed, and the mean reward and count variable for the chosen action are updated accordingly in line 10. The new mean reward and count variables are used to sample next prior values. After the κ-th round, the decision-maker invokes procedure `Eliminate` to eliminate actions or blocks of actions based on the calculated confidence scores and then zooms further.

The pseudo-code for `Eliminate` is provided in Algorithm 11. This procedure takes the discretized action set $\mathcal{A}$, the observed mean reward vector μ^t and the current round t as inputs. Initially, it checks whether the set of non-eliminated actions is large enough to be clustered or not. In each invocation of `Eliminate`, the algorithm reduces the block size, if possible, to zoom into the promising non-eliminated actions. To cluster actions, one can employ tree-based space-partitioning as in [168] or simple distance-based clustering algorithms like K-means [169]. In line 2 of `Eliminate`, the decision-maker calculates the lower and upper confidence bounds on the mean reward for each cluster. The bound estimation procedure estimates the lower and upper bounds of each cluster when there are multiple clusters, and for each action when only one cluster is present. A visual demonstration and worked-out example of the proposed blocking algorithm are given at the end of this section.

Our elimination is based on the observation that the algorithm will not benefit by exploring actions whose potential mean rewards are lower compared to alternative actions with high rewards. Based on this observation, we devise an elimination rule: *we eliminate the blocks whose upper confidence bound on the mean reward value is lower than the maximum of the lower confidence bounds on the mean reward.*

Algorithm 10: Recursive Block Elimination

Input : $\mathscr{X}, T, \kappa$, c

$\mathscr{A}$ = `Discretize`($\mathscr{X}$)
$\alpha = 1$, $\beta = 1$, $\xi = 1$
$\mathcal{C}$ = `cluster`($\mathscr{A}, c$)
Initialize(μ^0) `// initialise mean reward vector, e.g. 2.5`
Initialize(τ^0) `// initialise precision vector, e.g. 2`
`sample` μ^0 & τ^0 `according to Equation:` (6.1)
for $t = 1, 2, \ldots, T$ **do**
    $I_t = \arg\max_i \text{LogNormal}\left(\mu_i^{t-1}, \sqrt{\frac{1}{\tau_i^{t-1}}}\right)$
    `play arm` I_t `and observe reward` z_{I_t}
    `update mean reward and count for` I_t
    if $(t \% \kappa == 0)$ **then**
        `Eliminate`($\mathcal{C}, \mu^t, t$)
    `sample` μ^t & τ^t `according to Equation:` (6.1)
return action with highest mean

Algorithm 11: Eliminate Process

$u = |\mathcal{C}|$
$(lbs, ubs) =$ `GetConfidenceBounds`$(\mathcal{C}, \mu)$ `// estimate lower and upper bounds`
$maxlb = \max(lbs)$
for $\mathcal{S} \in \mathcal{C}$ **do**
    if *(*$u == 1$*)* **then**
        if *(*$|\mathcal{S}| == 1$*)* **then**
            return $\mathcal{S}$
        for $i \in \mathcal{S}$ **do**
            if *(*$ubs[i] \leq maxlb$*)* **then**
                eliminate action i from $\mathcal{S}$
        return $\mathcal{S}$
    if *(*$ubs[\mathcal{S}] \leq maxlb$*)* **then**
        $\mathcal{C} = \mathcal{C} \setminus \mathcal{S}$ `// eliminate action block` $\mathcal{S}$ `from` $\mathcal{C}$
return `cluster`($\mathcal{C}$)

6.3.2.1 Optimistic Initialization

In bandit algorithms, an effective heuristic to foster exploration involves setting initial values that are optimistically high, rather than starting at zero or a minimal number. For instance, consider a scenario where the rewards are modeled as coming from a zero-mean Gaussian distribution with a unit standard deviation; here, an initial value of 0.5 is significantly optimistic. This approach of setting higher initial values proactively encourages the algorithm to sample actions

that have not yet been explored. The rationale is that the initial high estimates make unexplored actions appear more attractive compared to those that have already been tested, which may have accumulated actual reward data that lowers their estimated values. To implement this strategy effectively, we initialize the mean and precision of the rewards with values that are tailored to the specifics of the problem at hand, ensuring that the exploration is both broad and aligned with the underlying reward distribution dynamics.

6.3.2.2 Updating Rewards of Non-observed Actions and Blocks

At each round, the decision-maker can only observe rewards from the action that was actually played. This limitation might lead to a biased sampling of actions since the mean and precision of actions that have never been played remain unchanged. To address this challenge, we begin by calculating the mean and precision for each block, based on the rewards of actions that have been played within that block. Subsequently, we adjust the mean and precision of the never-played actions within each block to match those of the block they are part of. For blocks where no actions have been played, we maintain the mean and precision at their initially set values, following the optimistic initialization technique described earlier. We ensure to update the mean and precision of all blocks at the end of each round to reflect the most recent data accurately. This method helps mitigate bias and promotes a more equitable representation of all actions in a block, irrespective of their play history.

6.3.2.3 Confidence Bound Estimation

In Algorithm 11, we estimate the maximum lower bound at line 3 and subsequently evaluate the upper bounds for each block or action. This process leads to the removal of blocks or actions that yield the least reward. A critical challenge in this approach is that once a block or action is eliminated, it is not reconsidered; thus, each elimination decision must be made with a high degree of confidence. To address this, we implement two confidence estimation schemes, each designed to ensure that eliminations are both justified and irreversible, minimizing the risk of prematurely discarding potentially valuable actions or blocks. Additionally, in our implementation, we apply a strict threshold of five samples from each cluster before deciding on confidence estimation and elimination. This means that any block of actions with fewer than five samples is automatically excluded from consideration for elimination. This threshold ensures that our confidence estimates are based on a robust sample size, and every action block is considered with equal importance.

Standard Error-based Bound Estimation: In [164], the authors proposed a standard error-based elimination given in Algorithm 12. The γ is a constant that measures the confidence. We set $\gamma = 2$ in our implementation, achieving $\sim 90\%$ confidence interval.

Algorithm 12: Error-based Bound Estimation

Input : $\mathcal{C}, \mu, t$
Initialize: γ
1 **for** $\mathcal{S} \in \mathcal{C}$ **do**
2 $\quad \mu_{\mathcal{S}} = \texttt{meanreward}(c, \mu)$ `// mean reward of block or action` $\mathcal{S}$
3 $\quad \sigma = \texttt{stderror}(\mathcal{S}, \mu)$ `// standard error`
4 $\quad lbs_{\mathcal{S}} = \mu_{\mathcal{S}} - \gamma \cdot \sigma$ `// lower confidence`
5 $\quad ubs_{\mathcal{S}} = \mu_{\mathcal{S}} + \gamma \cdot \sigma$ `// upper confidence`
6 **return lbs**, **ubs**

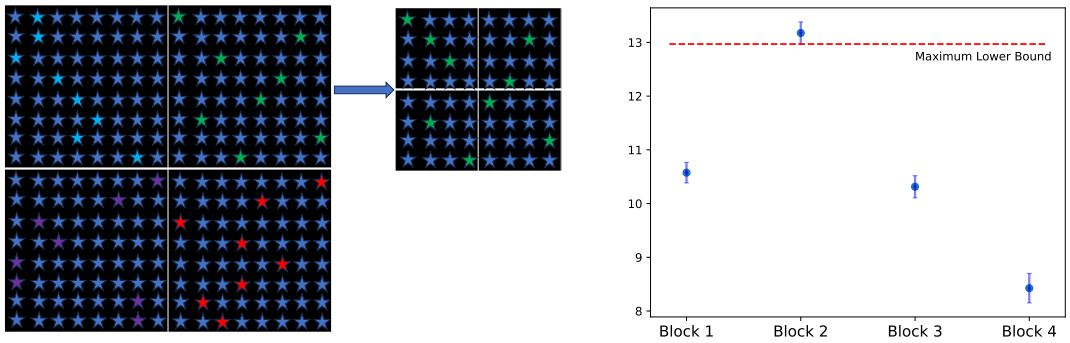
Algorithm 13: Hoeffding's Inequality-based Bound Estimation

Input : $\mathcal{C}, \mu, t$
1 **for** $\mathcal{S} \in \mathcal{C}$ **do**
2 $\quad \mu_{\mathcal{S}} = \texttt{meanreward}(\mathcal{S}, \mu)$ `// mean reward of block` $\mathcal{S}$
3 $\quad \sigma = \sqrt{\frac{\log t}{2 \times N_{\mathcal{S}}}}$ `// Hoeffding's Bound`
4 $\quad lbs_{\mathcal{S}} = \mu_{\mathcal{S}} - \sigma$ `// lower confidence`
5 $\quad ubs_{\mathcal{S}} = \mu_{\mathcal{S}} + \sigma$ `// upper confidence`
6 **return lbs**, **ubs**

Hoeffding's Inequality-based Bound Estimation: This estimation technique is based on Hoeffding's inequality, which is the basis for the UCB algorithm. The pseudo-code is given in Algorithm 13, where $N_{\mathcal{S}}$ is the number of observed samples from the block $\mathcal{S} \in \mathcal{C}$.

6.3.2.4 Top-two Sampling Extension

In the Top-Two probability sampling policy, the decision-maker randomly selects either the action deemed most likely to be optimal or the second most likely optimal action, based on the current posterior probabilities. This approach introduces a structured randomness into the selection process, allowing for a balance between exploring less certain options and exploiting the actions believed to offer the highest potential reward. By alternating between these two top choices, the policy aims to refine the accuracy of the estimated rewards while mitigating the risk of local optima, thus enhancing the overall strategy for decision-making within stochastic environments. A top-two extension to Thompson sampling has been proposed in optimal allocation problem settings with strong theoretical guarantees [158, 159]. We also consider the top-two version of our proposed algorithm given in Algorithm 10. In the top-two version of our algorithm, we replace the original sampling scheme (line 8) with equations: (6.3), (6.4) and (6.5). The rest of the algorithm remains the same.



(i) (ii)

Figure 6.1: Visual demonstration of the proposed algorithm. (i) initial blocks of actions and the remaining block after the elimination and (ii) visual representation of the upper and lower bounds of different blocks

$$J_1 = \arg\max_i \text{LogNormal}\left(\mu_{i,t-1}, \sqrt{\frac{1}{\tau_{i,t-1}}}\right) \tag{6.3}$$

$$J_2 = \arg\max_{i \neq J_1} \text{LogNormal}\left(\mu_{i,t-1}, \sqrt{\frac{1}{\tau_{i,t-1}}}\right) \tag{6.4}$$

$$I_t \sim \texttt{ with probability } \theta \texttt{ sample from } \{J_1, J_2\} \tag{6.5}$$

In Eq: (6.3) and Eq: (6.4), the algorithm finds the most likely optimal action and the second most likely action under the current posterior values, respectively. In Eq: (6.5), the algorithm randomly samples one of these actions, either the most likely optimal action or the second most likely option, with a probability of θ. The value of θ is often defaulted to 0.5, indicating equal probability for both actions.

6.3.2.5 Worked Out Example

To facilitate a better understanding of our algorithm, we provide a detailed illustrative example. In this scenario, we consider an action space represented in a two-dimensional plane as shown in Figure 6.1. As outlined in the algorithm description, we begin by discretizing the action space. The discretized actions are denoted by $\star$ symbols in Figure 6.1. We assume four clusters and organize the actions into four blocks, represented as square blocks divided by bold white lines in the figure. We then sample 32 actions from the action space (for illustration we made the sampling evenly spanning across all clusters), selecting eight actions from each block. The played actions are highlighted in cyan (Block 1), green (Block 2), purple (Block 3), and red (Block 4) colors. The rewards associated with these 32 actions are detailed in Table 6.1.

Blocks	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8
1	10.1	10.3	9.8	10.5	10.8	11	10.9	11.2
2	12.5	12.8	13.8	13	13.6	13.4	12.4	13.9
3	11.1	10.8	10.4	10.2	10	9.8	10.2	10
4	9.8	9.4	8.4	8.2	8.3	7.8	7.8	7.7

Table 6.1: Sampled revenues for the visual explanation of our algorithm

Blocks	*Mean*	*s.e*	LB	UB
1	10.575	0.171	10.233	10.917
2	13.175	0.206	12.763	13.587
3	10.313	0.206	09.901	10.725
4	8.425	0.274	07.877	08.973

Table 6.2: Mean, Standard Error, Lower and Upper Bounds of the four blocks

For elimination purposes, we calculate the empirical mean and standard error for each block, as described in Section 6.3.2.2. The mean and standard error of a block are calculated as the mean and standard error of all the rewards in that block, respectively. Subsequently, we compute the lower and upper bounds (in this example, we only consider standard error-based confidence score estimation, assuming $\gamma = 2$, as described in Section 6.3.2.3). The empirical mean, standard error, lower bound (LB), and upper bound (UB) of each block are provided in Table 6.2. This information is also visually represented in plot (ii) of Figure 6.1. According to our elimination rule, we eliminate all blocks with an upper bound lower than the maximum lower bound. The red dotted line in the plot signifies the maximum lower bound, and all blocks whose maximum upper bound lies below this line will be eliminated. Since the upper bounds of all blocks are lower than the maximum lower bound of 12.763, corresponding to Block 2, we eliminate all blocks except Block 2.

We then repeat the same procedure with the non-eliminated blocks. These non-eliminated blocks of actions are further clustered into four clusters, and sampling is continued to estimate the mean rewards. This process is repeated until the end of the trials or until there are not enough actions available to cluster. The plots in Figure 6.1 illustrates the further clustering of the non-eliminated blocks.

Remark 2 (Theoretical Analysis). We intend to investigate the theoretical properties of our algorithm in future work. Intuitively, the proposed block-elimination mechanism reduces the effective search space by eliminating groups of actions rather than individual actions, which may improve scalability in large action spaces. We also expect that sampling over κ rounds before elimination yields more reliable block-level estimates and helps avoid premature elimination of promising actions. Establishing such guarantees, however, requires new analytical tools beyond those used in existing elimination-based approaches.

6.4 Experimental Evaluation

In this section, we conduct a rigorous evaluation of our approach within real-world problem contexts. We focus on two dynamic assignment problems: *(i)* establishing optimal pricing strategies for parcel shipment services within logistics operations, and *(ii)* determining the most effective pruning ratios for Federated Learning (FL) applications. To accurately simulate these scenarios, we utilize synthetic datasets that are meticulously constructed based on real-world data. Our comprehensive evaluation involves a comparative analysis with leading elimination-based pure-exploration algorithms. Performance is assessed using metrics such as simple regret and the probability of misidentification. These synthetic datasets are intentionally designed to emulate the complexity and unique characteristics of the specific problem environments we are investigating.

In the logistics experiment, we consider the case $l = 2$, referring to two variants of a product/service. In the FL pruning experiment, we consider the case $l = 3$, referring to three variants of the product/service. In both cases, we used the K-means distance-based clustering [169]. The results strongly suggest that our block elimination algorithm is preferable to other sequential arm elimination algorithms in cases of a limited budget. We discuss the baselines and provide an in-depth analysis to verify the performance of our algorithm.

Note: The source code for the algorithms and data generation scripts can be found in the project page `https://github.com/shampp/tsrbe/`.

6.4.1 Performance Metrics & Baselines

Pure-exploration algorithms are compared using simple regret, while the misidentification probability is considered as an upper bound for the expected simple regret. The precise definitions of simple regret and misidentification probability can be found in Section 6.3.1. To gauge performance, we utilize *per-round* regret and misidentification probability as our key performance metrics. The per-round regret at round t is defined as the ratio of the cumulative simple regret to

the round. Mathematically, at round t, it is given as follows:

$$\frac{1}{t}\sum_{j=1}^{t} r_j, \tag{6.6}$$

where r_j is the simple regret at round j. We compare our algorithm against three state-of-the-art elimination-based algorithms: (i) Sequential Halving (SH) [153], (ii) Sequential Block Elimination (SBE) [146], and (iii) Variance Based Rejects (VBR) [164].

Sequential Halving (SH) Sequential Halving [153] assumes a discrete action set. The algorithm operates by estimating the mean rewards for *all* actions in phases, with each phase comprising a fixed number of rounds. At the conclusion of each phase, the SH method incrementally eliminates actions from the set until only one remains. Within each phase, actions are uniformly sampled from the remaining non-eliminated actions to estimate empirical mean values. The elimination criterion at the end of each phase involves removing the worst half of the actions based on empirical mean values.

Sequential Block Elimination (SBE) [146] proposed SBE algorithm. The SBE algorithm operates in phases, wherein blocks of actions are eliminated at the conclusion of each phase. SBE initiates the process by creating blocks of actions and sampling the central action of each block in every round. This design is grounded in the assumption that the central action represents the most connected action within each block. After the elimination round, the block with the lowest empirical mean reward for its central action is removed. While both SBE and our algorithm are block elimination methods, a fundamental distinction lies in SBE's approach of sampling *only* the central action from each block and making elimination decisions based solely on the mean reward of this action.

Variance Based Rejects (VBR) The VBR algorithm [164] eliminates actions based on upper and lower bounds on the mean rewards, following a similar procedure to that of SH and SBE algorithms. At the conclusion of each phase, the algorithm estimates the bounds and eliminates all actions whose upper bound is lower than the maximal lower bound. In the VBR algorithm, actions are uniformly sampled within each phase, and the bound estimation scheme aligns with the approach outlined in Algorithm 12.

Our algorithm variant is named **ReBEl**, which stands for **Re**cursive **B**lock **El**imination. The Top-Two sampling variant of this algorithm is referred to as **ReBEl (TT)**. To distinguish between different confidence score estimation schemes, we use the specific estimation technique as a subscript. For instance, **ReBEl** utilizing the **S**tandard **E**rror-based confidence estimation technique

is denoted as **ReBEl$_{\text{se}}$** and the version using **H**oeffding's **I**nequality-based technique is referred to as **ReBEl$_{\text{hi}}$**. This nomenclature allows for clear differentiation and easy reference within discussions and analyses following below.

Remark 3 (Limited Budget Setting). In the existing literature, all pure-exploration algorithms mandate that each action be tried at least once. Consequently, for demonstration purposes, we display the results for $T > k$ in the initial experiment. This condition ensures that the total number of trials T exceeds the number of actions k, aligning with the core requirements of pure-exploration methodologies and allowing us to effectively demonstrate the performance of our approach.

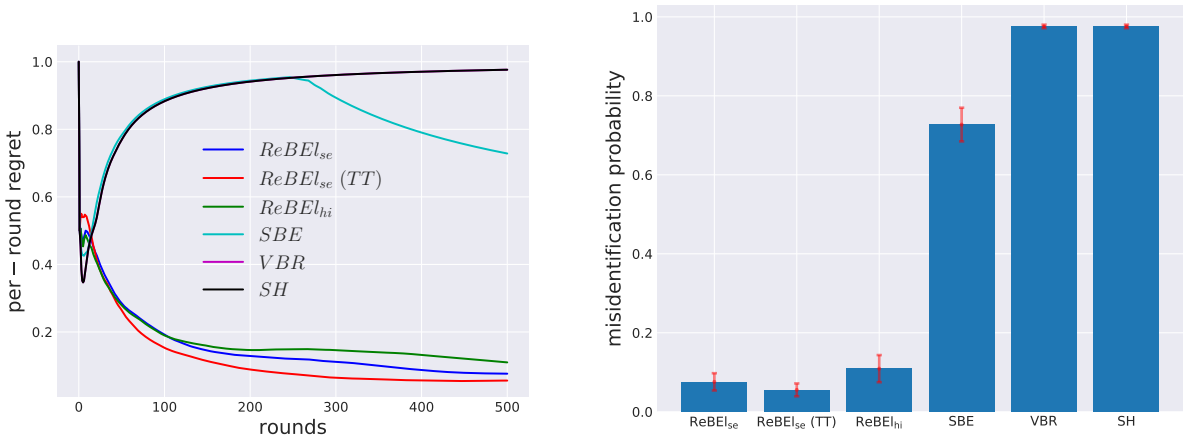
6.4.2 Case I: Dynamic Pricing for Logistics Operations

We initiated our evaluation by comparing the performance of our algorithm against baseline algorithms in a dynamic pricing scenario for logistics operations. The primary objective here was to determine the optimal prices for two variants of a logistic service to maximize total revenue. Specifically, we considered two types of logistic services ($l = 2$): *(i)* Standard and *(ii)* Express. For the Standard service, the minimum and maximum prices were set at 10 and 50, respectively. In contrast, the Express service prices ranged from 30 to 70 (in a specified monetary unit). We simulated the pricing dynamics of these 'standard' and 'express' shipping services using models based on real data from our industrial partner [170].

To address the dynamic pricing problem, we discretized the 2-dimensional action space into 256 distinct actions and utilized K-Means clustering to create action clusters, thereby facilitating the block elimination process. The revenue, or reward, is modeled as a joint function derived from the proportion of customers purchasing services at given price pairs (standard, express). This revenue function is nonlinear, depending on the proportions of customers who opt for the services at specified price levels. Each price pair is transformed into probability scores using the standard logistic function. We then randomly sample the proportions of customers purchasing the corresponding services. To foster a positive initial bias and encourage exploration, we set the initial mean rewards $\mu^0 = 2.5$ and the initial precision to $\tau^0 = 2$. This optimistic initialization helps in initially guiding the exploration towards potentially underexplored yet profitable actions.

Revenue function The revenue function for the dynamic pricing experiment is modelled based on a recent study using real data [7, 171] and is defined as:

$$r = p_1\lambda_1 + p_2\lambda_2, \tag{6.7}$$



(i) (ii)

Figure 6.2: Per-round regret and misidentification probability of different algorithms for the dynamic pricing problem.

where p_1 represents the probability of choosing the standard option, and p_2 is the probability of opting for the express service. Additionally, λ_1 and λ_2 denote the numbers of customers who chose options standard and express, respectively, based on the probabilities p_1 and p_2. The probabilities are further defined as follows:

$$\begin{aligned} u_2 &= \frac{1}{1+e^{(-\phi_2 a_2 - \psi_2)}} \\ u_1 &= (1-u_2)\frac{1}{1+e^{(-\phi_1 a_1 - \psi_1)}} \\ p_1 &= \frac{u_1}{u_1+u_2} \quad \text{and} \quad p_2 = \frac{u_2}{u_1+u_2} \end{aligned} \tag{6.8}$$

In the above, $\phi_1 = -0.05$, $\phi_2 = -0.05$, $\psi_1 = -0.5$, and $\psi_2 = 2$ are constants, whereas a_1 and a_2 denote the prices associated with the Standard and Express delivery services, respectively, with $a_1 \in [10, 50]$ and $a_2 \in [30, 70]$.

6.4.2.1 Results & Analysis

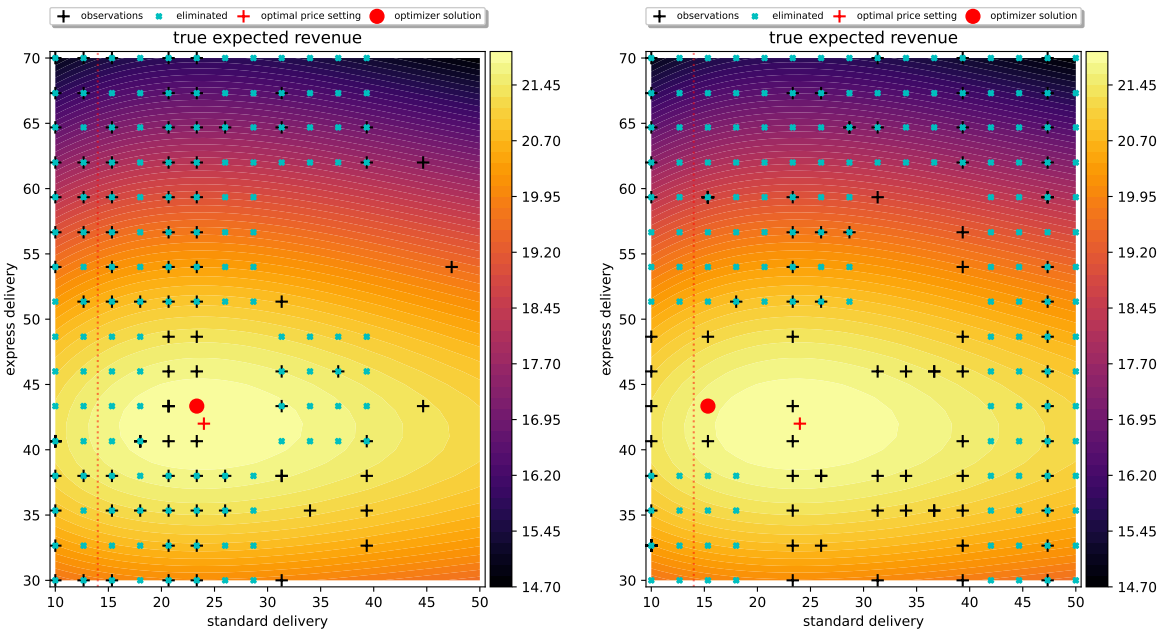
In Figure 6.2, we present the results of our first experiments (Case I). Figure 6.2 (i) & (ii) illustrate the average per-round regret and the misidentification probability of all algorithms over 50 random executions with a budget $T = 500$, respectively. For this experiment, we set the cluster size (c) to 16, the number of actions $k = 256$, and the number of rounds for the elimination $\kappa = 1.5 \times c = 24$. Our strategies exhibit a lower misidentification probability compared to other methods. Specifically, **ReBEl$_{\text{se}}$**, **ReBEl$_{\text{se}}$ TT**, and **ReBEl$_{\text{hi}}$** achieved the lowest misidentification probabilities and per-round simple regret. It is notable that among our proposed strategies, **ReBEl$_{\text{se}}$** performed marginally better than **ReBEl$_{\text{hi}}$**. This suggests that standard error provides

a slightly more reliable confidence score than Hoeffding's inequality for this set of price distribution data. However, this is not universally the case, as will be demonstrated in our second set of experiments. As highlighted by theoretical studies such as those by [158] and [159], Top-Two sampling achieved results that were marginally better than the conventional approach. Similar to the case of different confidence score estimation techniques, the significance of this result is not pronounced. This indicates a need for further experimentation to understand the nuances of these methods under varying conditions.

Among the block elimination strategies, it is clear that the performance of Sequential Block Elimination (SBE) does not match that of our proposed algorithms. The effectiveness of SBE is significantly influenced by the number of clusters and the geometry of the optimal region. In the dynamic pricing experiment, the performance of SBE notably benefits from a lower number of clusters, as demonstrated in the sensitivity analysis presented in Section 6.4.4. This improvement is primarily due to the substantial influence the geometry of the optimal region has on the overall performance of SBE. The optimal or near-optimal price region of logistic revenue distribution is relatively expansive, meaning that a smaller number of clusters ensures that optimal or near-optimal prices are close to the center of each cluster. Conversely, a larger number of clusters may lead to a fragmentation of this optimal region, resulting in degraded performance. This dynamic can be visualized and further explained using the contour plot shown in Figure 6.3.

In this contour plot, the shipping price revenue function displays a smooth gradient, leading to only minor changes in revenue values around the optimal points, depicted by the innermost yellow ellipse. The color intensity exhibits minimal variation when moving from the inner ellipse to the outer ellipses. In our experiment, we initially utilized SBE with 16 clusters, which resulted in considerable time spent estimating optimal rewards due to the wide dispersion of price points within each cluster. Once the optimal region was accurately identified, SBE began recommending optimal actions, as indicated by the decreasing per-round simple regret starting around round 300. However, our findings suggest that for our specific set of pricing data, SBE achieves better performance with fewer clusters (see Section 6.4.4). This configuration allows SBE to more swiftly locate the optimal prices, although it still underperforms compared to our newly proposed methods. This analysis underscores the need for careful cluster selection based on the characteristics of the pricing distribution to optimize the efficiency and effectiveness of the SBE strategy.

As anticipated, both SH (Sequential Halving) and VBR (Variance Based Rejects) exhibited poor performance in our experiments. This outcome is largely due to the significant number of rounds these strategies devote to exploring sub-optimal arms before deciding to eliminate them. The results underscore that in scenarios with a limited budget, strategies that attempt to try all actions are sub-optimal. Instead, block elimination strategies have proven more effective. Among our three proposed block-elimination strategies- **ReBEl$_{\mathbf{se}}$**, **ReBEl$_{\mathbf{se}}$ TT** and **ReBEl$_{\mathbf{hi}}$**-



(i) (ii)

Figure 6.3: Contour map of the action space and the running snapshot of **ReBEl$_{se}$** (i) & **ReBEl$_{hi}$** (ii) after the end of 100 rounds (Dynamic Pricing problem).

the optimal action is identified within fewer than 100 rounds of the allocated budget. This efficiency contrasts sharply with that of SBE, which required close to 300 rounds, and other elimination strategies such as SH and VBR, which took over 500 rounds to locate the optimal action.

These findings highlight the efficiency of block elimination strategies in quickly converging to optimal actions, particularly in settings where resource constraints necessitate rapid decision-making. This comparative analysis clearly illustrates the advantages of the block elimination approach over more traditional methods, emphasizing its utility in optimizing performance within limited experimental budgets.

In addition, we present contours and snapshots of **ReBEl$_{se}$** and **ReBEl$_{hi}$** after the completion of a hundred rounds ($T = 100$) in Figure 6.3. These visuals effectively illustrate how our algorithms manage to eliminate sub-optimal blocks of actions, regardless of the size and shape of the optimal region, in comparison to baseline strategies. Specifically, the plots showcase the *ground truth* optimal pair of solutions and the achieved optimal solution by **ReBEl$_{se}$** and **ReBEl$_{hi}$** after the conclusion of 100 rounds, i.e., the budget. Actions that are marked in cyan colour represent those that have been eliminated, while those marked with a '+' symbol indicate actions that have been played. Notably, both **ReBEl$_{se}$** and **ReBEl$_{hi}$** are successful in eliminating most suboptimal actions within the 100-round budget and consistently recommend actions that are closest to the optimal solution. Furthermore, these visuals provide insights into the per-

formance of SBE. With only 16 clusters in this instance, the centroids often lie distant from the optimal region, contributing to the performance degradation of SBE. The sensitivity analysis further reveals that SBE performs better with a reduced number of clusters. The overall results underscore that, in scenarios with limited budgets where the budget is small compared to the number of actions ($T < k$), our algorithm consistently outperforms state-of-the-art algorithms. This demonstrates the robustness and efficiency of our block elimination strategies in optimizing the selection process under resource constraints.

6.4.3 Case II: Dynamic Pruning in Federated Learning

In our second experiment, we evaluated the performance of our algorithms against baseline methods in a three-dimensional problem setting focused on determining the optimal pruning ratios in Federated Learning (FL). FL techniques are instrumental in addressing privacy regulations and communication bandwidth limitations, facilitating distributed model training from decentralized data sources [172]. In FL, data are collected at edge nodes (e.g., smartphones) and used to update local models, which are then sent to a central server node for aggregation into a global model. This global model is subsequently shared among the edge nodes [172]. Given that typical deep learning models consist of millions of parameters, distributing, storing, or processing these models in their entirety at individual nodes is impractical. A prevalent approach in FL is to distribute a pruned version of the global model among the nodes to alleviate these challenges [173]. The heterogeneity of the nodes in FL—varying in computational power, storage capacity, and network connectivity — means that the optimal pruning ratio differs across nodes. Dynamic pruning, therefore, aims to tailor the pruning ratios to the specific needs and capabilities of each node to optimize performance and efficiency [132, 173].

In our simulation of the Federated Learning (FL) scenario, we structured a network that includes a server node and three heterogeneous nodes connected to the server. With $l = 3$ (representing the three client nodes), our experimental actions are modeled in the three-dimensional space $[0,1]^3$, reflecting a range of possible pruning ratios for each node. The computational and communication gains from each node in this network are quantified using a joint distribution model, where the gains for individual nodes are described by Poisson distributions based on the number of successful packets processed by the nodes. Each experimental round computes the mean of twenty-five samples, which correspond to taking twenty-five samples of the communication and computational gains for the same set of pruning ratios. This approach parallels the methodology used in our dynamic pricing experiment (Case I), where the parameter represented the number of customers. In the sensitivity analysis, we delve into how this parameter — the number of samples per round—affects the performance of our proposed algorithms. In this experiment, we aim to demonstrate that our algorithms can adaptively optimize the pruning ratios in real-time, leading to significant improvements in both communication efficiency and computational workload distribution across the nodes.

We discretized the action space into 4096 action triplets in the 3-dimensional plane and applied K-Means clustering to create 32 clusters. Considering the significant increase in the number of potential actions in this experiment compared to the dynamic pricing setup, we adapted our methodology to employ block elimination algorithms with a budget of $T = 1000$. Although this budget is notably larger than that of the previous experiment, the action space remains large relative to the available budget, with $k = 4096$ and $T = 1000$. As a result, non-block-elimination-based strategies such as SH and VBR were not included as baselines for this experiment. These methods require a proportionally higher budget relative to the number of actions to be effective, making them less suitable under the budget constraints of this experiment. As in the dynamic pricing experiment, the results we present are based on averages over 50 random executions to ensure robustness and minimize the impact of outliers or anomalous runs. To initiate the experiment under conditions that favor optimistic exploration, we set the initial bias parameters to $\mu^0 = 1.3$ and $\tau^0 = 2$. These values were selected empirically to encourage optimistic exploration and were kept fixed across all runs to ensure a consistent experimental setting. These parameters influence the initial degree of optimistic exploration and therefore have their greatest effect during the early rounds of the algorithm. As additional reward observations are collected, the influence of the initial prior is progressively reduced by the data-driven parameter updates. Since μ^0 and τ^0 were kept fixed throughout the experiments, the thesis does not make an empirical claim regarding sensitivity to alternative prior values; instead, the reported comparisons evaluate all algorithms under the same initial prior conditions.

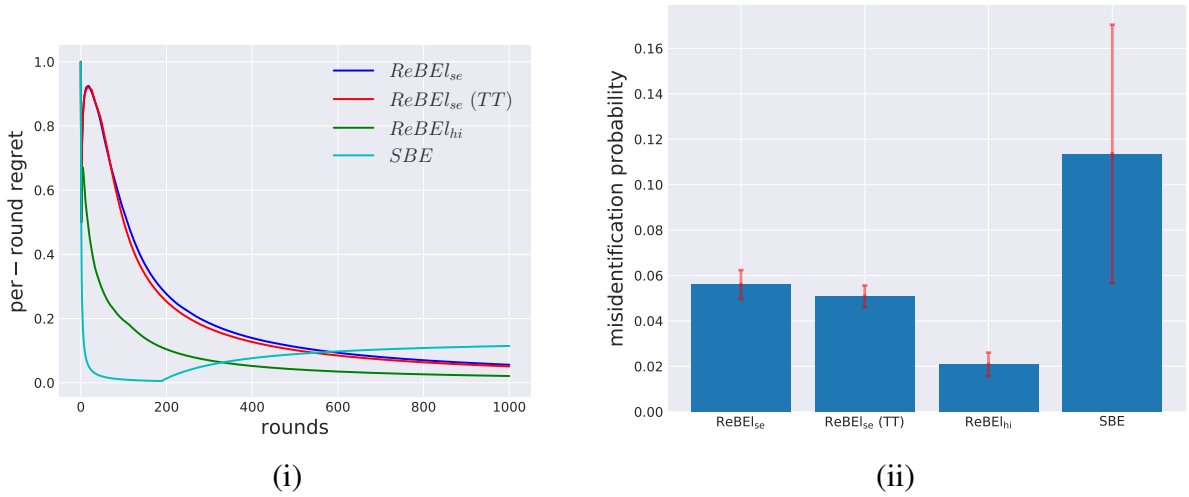
Reward Function Like in the dynamic pricing experiment, we model the reward function based on a recent study [7, 171]. The reward function is defined as follows:

$$r = \frac{\lambda_1 p_1 + (1-p_2)p_1\lambda_2 + (1-p_3)(1-p_2)p_1\lambda_3}{p_1 + (1-p_2)p_1 + (1-p_3)p_2} \tag{6.9}$$

In Equation (6.9), λ_1, λ_2 and λ_3 are the Poisson distributed random variable sampled according to the probabilities p_1, p_2 and p_3, respectively. The probabilities are defined as functions of pruning ratios:

$$\begin{aligned} p_1 &= 0.6 - |a_1 - 0.6| \\ p_2 &= |a_2 - 0.7| \\ p_3 &= |a_3 - 0.5| \end{aligned} \tag{6.10}$$

In Equation (6.10), a_1, a_2 and a_3 are the chosen action triplet, i.e., the pruning ratios for nodes 1, 2 and 3, respectively.



(i) (ii)

Figure 6.4: Per-round regret and the misidentification probability of different block-elimination algorithms for the 3D dynamic pruning problem

6.4.3.1 Results & Analysis

The results of our pruning experiments in the context of FL are detailed and visualized in Figure 6.4. In this figure, (i) illustrates the per-round regret, and (ii) depicts the misidentification probability. The curves correspond to the average per-round regret computed over 50 independent runs; no additional smoothing was applied. The different hyperparameters are set as follows: the number of actions at $k = 4096$, the number of clusters (c) at 32, and the number of rounds for elimination $\kappa = 1.5 \times c = 48$.

Observing the plots, it is evident that despite the relatively large action space compared to the budget (4096 against 1000), the block elimination strategies successfully eliminated suboptimal actions and frequently recommended optimal or near-optimal actions. Notably, in this experiment, the Hoeffding's inequality-based elimination strategy outperformed the standard error-based elimination strategy, though marginally. As in the case of dynamic pricing, the Top-Two sampling version demonstrated better performance than the vanilla version. The observed performance improvement, though marginal, with the Top-Two sampling version in both the dynamic pricing and the FL pruning experiments underscores a significant trend. This trend suggests that the Top-Two sampling strategy consistently outperforms any vanilla probabilistic sampling approaches in scenarios focused on pure exploration. This observation is in line with the theoretical underpinnings.

It is noteworthy that **ReBEl$_{\mathbf{se}}$**, **ReBEl$_{\mathbf{se}}$ TT**, and **ReBEl$_{\mathbf{hi}}$** consistently resulted in lower per-round regret and misidentification probability compared to SBE. While the performance of SBE is better than in the dynamic pricing experiments, particularly in the initial rounds, its superiority diminishes as the number of rounds increases. As depicted in the per-round regret plot, our proposed algorithms start yielding superior results after around 200 rounds.

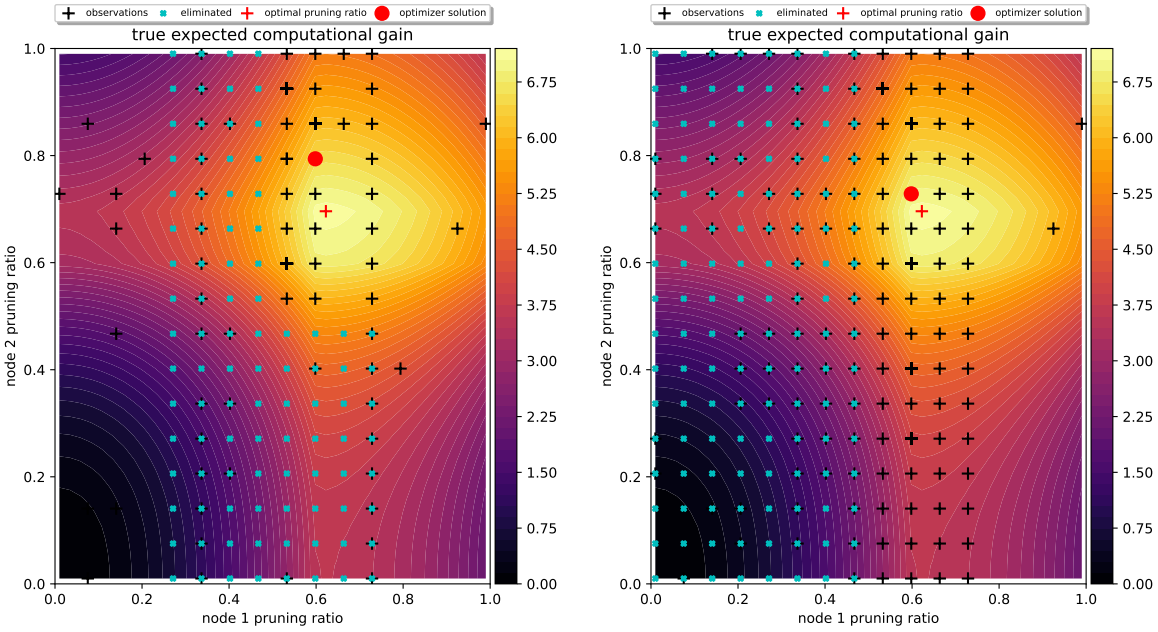
The noticeable improvement in the performance of SBE in this experiment can indeed be attributed to the increased number of clusters and the specific characteristics of the optimal region. Our sensitivity study highlights that the effectiveness of SBE is significantly influenced by the number of clusters it utilizes. A larger number of clusters allows for finer segmentation of the action space, which can be particularly beneficial when the optimal region is narrow, as it increases the likelihood that one of the clusters is closely aligned with this optimal region. Additionally, the influence of the shape of the optimal region on SBE's performance is critical. As illustrated in the contour plot of the 2D projected action space in Figure 6.5, the optimal action region in this experiment is relatively narrow, especially when compared to the broader, more dispersed optimal regions observed in the logistics experiment. In such settings, where the optimal region is narrow, having a larger number of clusters ensures that the central action within each cluster is more likely to be near the optimal actions. This proximity can significantly enhance the ability of the algorithm to zero in on the most promising actions quickly. This configuration — larger number of clusters combined with a narrowly defined optimal region—helps explain why SBE performed notably better in this scenario. It effectively demonstrates how the structural aspects of the problem, such as the distribution and granularity of the action space, can directly impact the suitability and success of different baseline strategies.

As previously mentioned, SH or VBR were not engaged in this experimental setting, as these algorithms require at least the same number of rounds as the number of actions. Our focus here is specifically on cases where the budget is very limited. The results strongly reinforce the notion that in limited budget cases, where the budget is small compared to the number of actions $(T < k)$, our algorithm consistently outperforms state-of-the-art algorithms.

In addition, we have included snapshots of **ReBEl$_{\text{se}}$** and **ReBEl$_{\text{hi}}$** after the completion of 200 rounds ($T = 200$) in the contour plot of the 2D projection of the original problem, (see Figure 6.5). Both algorithms successfully recommended the optimal action within the allocated budget of 200 rounds. The snapshots vividly illustrate the elimination strategies employed by each algorithm. **ReBEl$_{\text{hi}}$** efficiently eliminated the entire region of sub-optimal actions lying to the left of the optimal action, while **ReBEl$_{\text{se}}$** eliminated both the left and bottom halves.

6.4.4 Sensitivity Analysis

In this section, we carry out the sensitivity analysis of the algorithms against two hyper-parameters of the proposed algorithms and the number of samples collected in each round. The two hyper-parameters of the algorithms are: the number of clusters c and the number of rounds to the elimination phase κ. The number of clusters impacts the performance of all block elimination-based algorithms: **ReBEl$_{\text{se}}$**, **ReBEl$_{\text{se}}$ TT**, **ReBEl$_{\text{hi}}$**, and SBE, whereas the number of rounds to the elimination phase κ affects only **ReBEl$_{\text{se}}$**, **ReBEl$_{\text{se}}$ TT**, and **ReBEl$_{\text{hi}}$**.



(i) (ii)

Figure 6.5: Contour map of the action space, and the running snapshot of **ReBEl$_{\text{se}}$** (i) & **ReBEl$_{\text{hi}}$** (ii) after the end of 200 rounds (2D projection of the original dynamic pruning problem).

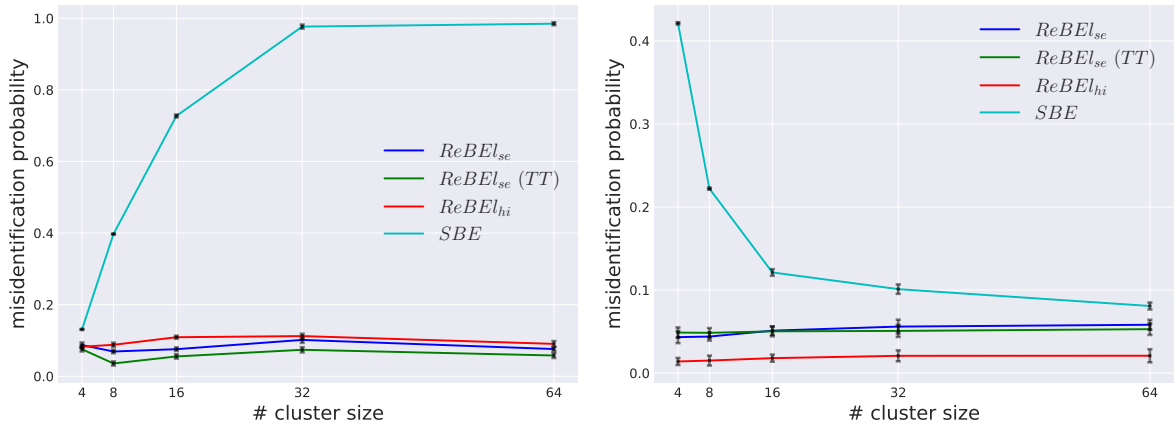
In the initial sensitivity analysis, we examine the impact of varying the number of clusters on misidentification probability, an upper bound on simple regret. This relationship is illustrated in Figure 6.6, where Figure 6.6 (i) depicts the performance of the dynamic pricing experiment and Figure 6.6 (ii) depicts the performance of the dynamic pruning experiment. In the context of logistic services, the performance of our proposed algorithms—**ReBEl$_{\text{se}}$**, **ReBEl$_{\text{se}}$ TT**, and **ReBEl$_{\text{hi}}$**—demonstrates minimal variation with an increase in the number of clusters. It is noteworthy that for smaller cluster sizes, specifically when the number of clusters equals 4 and 8, there is a slight decrease in misidentification probability compared to scenarios with larger numbers of clusters. This increment is marginal but indicative of the sensitivity of the algorithm to cluster size at lower levels. Surprisingly, further increases in cluster size from 16 to 32 and 64 do not result in substantial improvements in misidentification probability. This trend can be attributed to the limited size of the action set, which suggests that the algorithms are capable of identifying the optimal action from a relatively small set of clusters. Consequently, there is no significant advantage in augmenting the number of clusters beyond a certain point. These findings are consistent with those observed in the original dynamic pricing experiments, suggesting that our algorithms are robust across different cluster configurations.

The misidentification probability plot for the SBE algorithm clearly indicates that its performance is significantly influenced by the number of clusters, even when the action set is small. Notably, with a smaller number of clusters (e.g., 4, as illustrated in the plot), the misidentification probability for SBE is on par with that of our proposed methods. However, as the number of clusters increases, the performance of SBE deteriorates markedly, with the misidentification probability approaching one. This outcome is primarily due to the geometric characteristics of the optimality region specific to this problem instance. In the scenario of dynamic pricing, where the optimality region tends to be smoother and larger, having a smaller number of clusters is advantageous. This smaller cluster configuration ensures that the centroid action, which represents the average position of the actions within a cluster, is likely to be closer to the optimal region. Consequently, fewer clusters can be more effective for ensuring that SBE performs near optimally, as it reduces the complexity and variability within the action set, facilitating closer approximations to the optimal action.

A similar trend is evident in the dynamic pruning experiment, where the **ReBEl$_{\text{hi}}$** algorithm outperforms both **ReBEl$_{\text{se}}$** and **ReBEl$_{\text{se}}$ TT**, even with a smaller number of clusters. A critical observation from the plot in Figure 6.6 is that the performance of SBE aligns with that of the **ReBEl$_{\text{se}}$** algorithm as the number of clusters increases. This differs from the trend observed in the dynamic pricing results, where the performance of SBE deteriorated with more clusters. The key to understanding this variance lies in the geometry of the optimality region. In the dynamic pruning scenario, the optimality region is relatively small, and increasing the number of clusters enables SBE to better approximate the optimal action, enhancing its performance. Essentially, the effectiveness of the SBE algorithm is closely linked to the geometry of the optimality region and the number of clusters involved. In contrast, our proposed methods—**ReBEl$_{\text{hi}}$**, **ReBEl$_{\text{se}}$**, and **ReBEl$_{\text{se}}$ TT**—exhibit a robust performance that is less influenced by these factors. This resilience highlights their suitability for a broader range of scenarios where the shape and size of the optimality region may vary significantly.

In the second sensitivity analysis, we investigated the robustness of the algorithm against the number of rounds in the elimination phase, denoted as κ. We plotted the misidentification probability against κ in Figure 6.7. Figure 6.7 (i) depicts the performance of the dynamic pricing experiment, while Figure 6.7 (ii) depicts the performance of the dynamic pruning experiment. In both experiments, we set the number of rounds in the elimination phase to be a multiple factor of the initial number of clusters (c). In the case of dynamic pricing, we set the number of clusters to 16, and in the case of dynamic pruning, we set the number of clusters to 32. Since κ does not impact the performance of the SBE algorithm, we exclude SBE from this set of experiments.

The analysis reveals that the misidentification probability does not follow a specific trend as a function of the κ value. This observation suggests a nuanced relationship between elimination rounds and algorithm performance. Our hypothesis posits that keeping the number of rounds to the elimination phase too small is not recommended, as this premature elimination can degrade



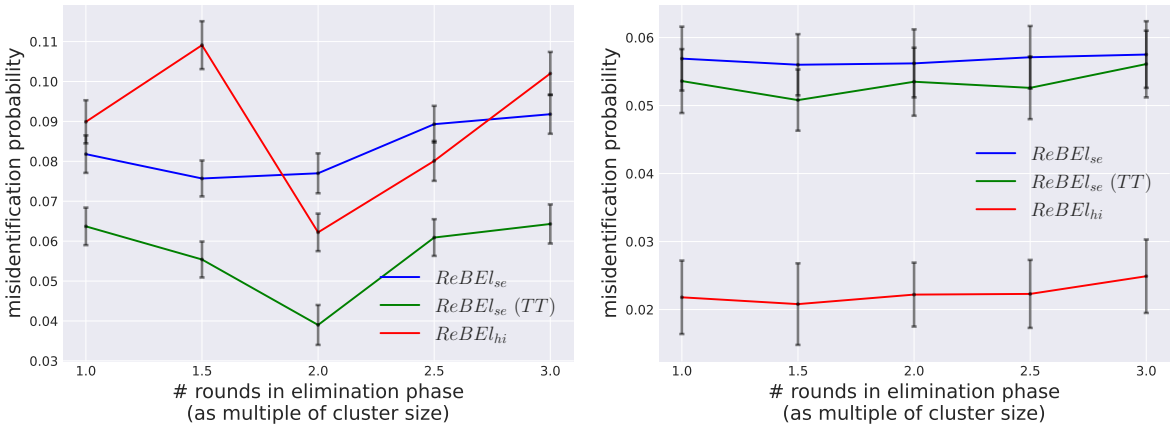
(i) (ii)

Figure 6.6: Misidentification probability as a function of the number of clusters in (i) Dynamic Pricing & (ii) Dynamic Pruning experiments.

the performance of all elimination-based algorithms because a limited number of samples might not provide sufficient information to make accurate decisions about which actions to eliminate. An elimination decision based on very few samples can result in the elimination of potentially 'good' blocks. Conversely, excessively extending the number of rounds in the elimination phase does not notably enhance performance. This is because prolonging the elimination process tends to include too many suboptimal or 'bad' actions, leading to inefficiencies in the algorithm.

For algorithms like **ReBEl$_{\text{se}}$** and **ReBEl$_{\text{se}}$ TT**, we observe that misidentification probability initially decreases as the number of samples in the elimination phase increases. This trend continues up to a certain threshold, beyond which the misidentification probability begins to increase. This pattern aligns with our hypothesis and underscores the existence of an optimal balance in the number of samples required during the elimination phase to maximize performance effectively. In the case of the **ReBEl$_{\text{hi}}$** algorithm, no consistent trend is discernible in terms of κ value adjustments. However, in general it appears that setting the κ value to approximately 1.5 to 2 times the initial cluster size tends to yield the best results.

In our final sensitivity analysis, we examine how the number of samples collected in each round impacts the performance of the proposed algorithms. This study is critical in understanding the reliability of the estimates, which directly influences the effectiveness of the algorithms in identifying optimal actions and eliminating suboptimal actions.



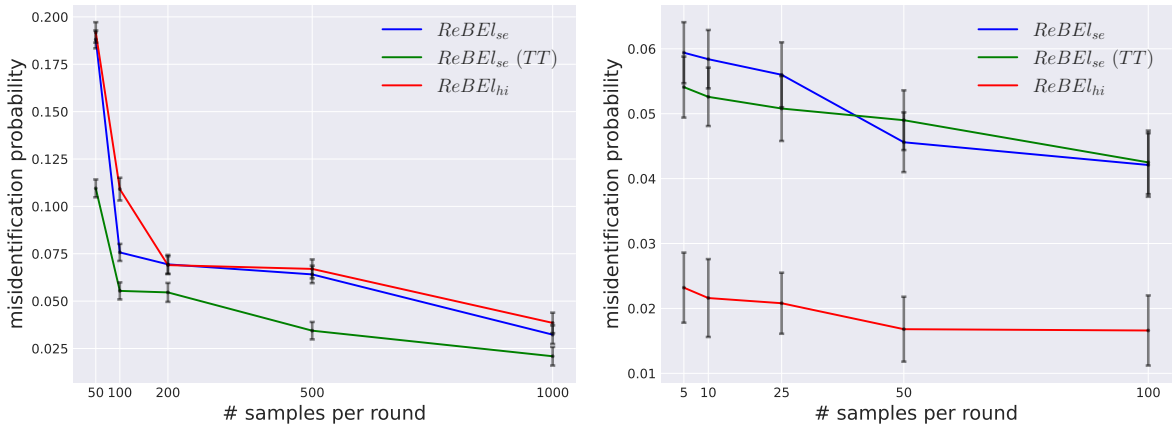
(i) (ii)

Figure 6.7: Misidentification probability as a function of the number of rounds in the elimination phase in (i) Dynamic Pricing & (ii) Dynamic Pruning experiments.

In the dynamic pricing scenario, each round involves presenting prices to 100 customers. The revenue generated from each price pair during that round is averaged to estimate the revenue for that particular price setting. It is reasonable to suggest that increasing the number of customers per round would enhance the reliability of the revenue estimation. A larger sample size tends to reduce the variance in the estimate, thereby providing a more accurate reflection of the true mean revenue. This enhanced accuracy can significantly aid the algorithm in swiftly identifying the most lucrative price points and discarding less profitable options.

In the dynamic pruning case, pruning ratio triplets are tested 25 times on the same network in each round, with the average network gain being calculated subsequently. Similar to Case I, increasing the number of trials per round would presumably lead to a more accurate estimation of the mean network gains. This is because a greater number of trials diminishes the impact of anomalies or outliers on the average gain, leading to a more stable and reliable estimation.

In both cases, a higher number of samples per round allows for a more robust and accurate estimation of mean rewards, which is vital for the algorithms' efficiency. More reliable estimates help the algorithm distinguish more effectively between the performance of different blocks or actions, facilitating faster identification of the optimal blocks and more accurate elimination of the suboptimal ones. This can be observed in the plots in Figures 6.8 (i) and (ii), where the misidentification probability is plotted against the number of customers (dynamic pricing) and the trials per round (dynamic pruning), respectively.



(i) (ii)

Figure 6.8: Misidentification probability as a function of the number of customers/trials sampled per round in (i) Dynamic Pricing & (ii) Dynamic Pruning experiments.

6.5 Conclusions

We propose a class of Thompson sampling-based recursive block elimination algorithms for dynamic assignment in a pure-exploration setting with a limited budget. Our algorithm begins by discretizing the continuous action space into a set of finite discrete actions, followed by the recursive elimination of suboptimal action blocks. Additionally, we implement a zooming mechanism that further clusters these action blocks recursively and eliminates suboptimal blocks. This elimination process involves calculating confidence bounds over the blocks. We have proposed two distinct techniques for estimating these confidence bounds. Moreover, we have explored variants of the proposed algorithms that employ top-two sampling. We conducted extensive experiments across two problem settings, demonstrating that our approach achieves a lower misidentification probability compared to state-of-the-art algorithms. We also examined the sensitivity of the algorithms to various hyperparameters. Our sensitivity analysis indicates that the proposed algorithms are robust across a range of hyperparameter settings.

For future work, we plan to conduct a regret analysis of the proposed algorithms and explore their extension to adversarial settings.

Chapter 7

Discussion and Conclusions

This thesis analyses the opportunities and challenges associated with intelligent and adaptive decision-making for service management in Edge Computing (EC) settings. The traditional paradigms of centralised computing are facing increasing constraints as the creation of data moves closer to the devices used by end users and as the desire for low-latency, high-efficiency systems increases. This study provides decision-theoretic and learning-based frameworks as a solution to this challenge. These frameworks enable edge nodes to make timely and autonomous decisions about service replication, task offloading, and node selection. These choices are made without relying on central control or static heuristics. Furthermore, this thesis emphasises the need to connect theoretical models with real deployment difficulties. The study advances the understanding of how edge systems might achieve resilience, scalability, and robustness under unpredictable and challenging settings by offering adaptive frameworks that can learn and change in real time. The proposed approaches may also be applied in various distributed systems besides EC, such as automobile networks and 6G architectures, making them extremely adaptable.

7.1 Key Contributions

The contributions of this thesis address five research problems that are interconnected with one another. Each of these questions focuses on a different difficulty that is present in decentralised edge configurations. Together, these contributions constitute a cohesive pipeline of decision-making intelligence for EC nodes. From replication and offloading to orchestration and adaptive node selection, this thesis shows how theoretical ideas from decision theory and machine learning can be implemented systematically to real-world EC problems.

7.1.1 Time-Optimized Decisions for Service Replication and Offloading

This is the first contribution that addresses the topic of how an edge node may independently determine whether to replicate a service that is absent or to offload a task. This decision is made despite the uncertainty in future demand. The suggested solution refers to a framework for sequential decision-making that is founded on the concept of **Optimal Stopping Theory (OST)**. This framework approaches service replication as an issue that involves time. A 1-stage look-ahead rule is used by the edge node in order to monitor the arrival of requests and ascertain the most appropriate moment to take action. The results of experiments conducted in synthetic settings motivated by smart city applications suggest that the model has the capability to decrease the amount of time it takes for replication to occur, improve service availability, and increase overall system responsiveness.

In addition to enhancing latency and availability, the OST-based strategy provides a mechanism for balancing trade-offs between early replication (with likely resource waste) and delayed replication (with potential service degradation). This balance allows the framework to be used in a wide range of applications, including video streaming, emergency response services, and real-time data analysis, all of which need accurate timing. The model's adaptability implies that it might be used in combination with predictive caching or proactive service placement strategies in the future.

7.1.2 Cost-Aware Orchestration under Time-Varying Demand

Concerns about the orchestration of edge services in contexts where the volume of task requests decreases over time were the subject of the second significant difficulty. This thesis presents the **Discounted Optimal Stopping (DOST)** framework, which combines decaying arrival patterns and delayed offloading costs. This framework is in contrast to standard models, which use static or uniform demand assumptions. Through the description of arrival rates as Poisson processes that decrease exponentially, DOST makes it possible for edge nodes to dynamically alter their judgments in response to an ever-changing demand. The results of simulations demonstrate that DOST is superior to other methods in terms of properly managing resources in non-stationary situations and maximizing long-term reward. Another feature of DOST is its ability to accommodate different cost structures across diverse nodes. By prioritising services with higher demand and reducing unnecessary service replication and task offloading, the proposed approach may also contribute to reducing overall system energy consumption and monetary expense. Furthermore, DOST's flexible nature makes it suitable for sectors with fluctuating demand and limited resources, such as IoT-enabled agriculture, industrial automation, and energy-sensitive sensor networks.

7.1.3 Expert-Driven Task Placement with Robust Performance Guarantees

By including an **expert-driven** selection process into the MAB framework, this contribution improves the accuracy of decision-making. This improvement is achieved by building upon the previous node selection technique. In the above procedure, each expert is represented by a deep learning model—specifically, **Feedforward Neural Networks (FNN)**, **Recurrent Neural Networks (RNN)**, and **Long Short-Term Memory (LSTM)** networks—that serve as autoregressive oracles predicting node behaviour based on historical system feedback, such as heartbeat signals and resource usage.

The technique that has been provided takes into account the historical accuracy of each expert in order to determine which model is the most trustworthy at any given moment. Gradient bandit learning is then used in order to update the probability of node selection appropriately. By using this dual-layer selection technique, the system is able to successfully adjust to variations that occur in real time while simultaneously preserving steady convergence. The extended ExpGradBand model delivers reduced cumulative regret and quicker convergence in a range of adversarial, non-stationary, and noisy situations. This is in comparison to typical MAB algorithms such as Thompson Sampling, Exp3, and Exp4, which are traditional MAB algorithms.

This work highlights the advantages of combining model-based and model-free learning. By employing deep learning specialists while keeping an active adaptation layer, the architecture strikes a compromise between predictive strength and flexibility. This hybrid approach may be applied to additional scenarios involving several predictive oracles, such as traffic forecasting, healthcare monitoring, and predictive maintenance in industrial systems.

7.1.4 Node Selection in Dynamic and Adversarial Environments

In contexts where edge nodes vary in capacity, availability, and feedback dependability, the fourth research question concerns task allocation. In order to solve this issue, the thesis presents **ExpGradBand**, which is an innovative expansion of the Multi-Armed Bandit (MAB) architecture that incorporates contextual learning and gradient-based adaptation. It is possible for the system to react in an adaptive manner even when feedback signals are delayed, noisy, or nonexistent, since ExpGradBand dynamically picks both a node and a decision method based on real-time observations. This method dramatically reduces the amount of cumulative regret and improves the resilience of decisions, making it suitable for highly dynamic decision-making environments.

The experimental evaluation also assesses the performance of ExpGradBand in terms of per-round regret under dynamic and adversarial settings, providing empirical evidence of its effectiveness across a range of scenarios. Moreover, ExpGradBand adds flexibility by allowing decision-makers to inspect how the system balances various tactics over time. This opens up the possibility of expanding the concept to federated or cooperative edge systems, where nodes might share partial information to improve dependability even further.

7.1.5 Pure-Exploration under Budget Constraints

The fifth research topic addresses dynamic assignment problems under strict budget constraints, where exhaustive exploration of large or continuous action spaces is infeasible. To this end, Chapter 6 introduces a Thompson Sampling-based Recursive Block Elimination framework that efficiently balances exploration and confidence-driven elimination. By recursively partitioning the action space and discarding sub-optimal regions, the proposed approach concentrates the available sampling budget on promising areas while maintaining probabilistic guarantees on identification accuracy.

This framework is particularly well suited for resource-constrained decision-making scenarios, such as dynamic pricing and federated learning pruning, where only a limited number of evaluations can be afforded. The experimental results demonstrate reduced misidentification probability and improved simple regret compared to state-of-the-art elimination-based methods, highlighting the scalability and practical relevance of the proposed solution.

7.2 Broader Implications and Limitations

Through the incorporation of autonomy, flexibility, and foresight into EC nodes, the solutions that are offered in this thesis contribute to the advancement of the frontier of edge intelligence. The study offers a realistic approach for allowing low-latency, decentralized, and resource-efficient service delivery in situations such as smart cities, industrial Internet of Things, and autonomous networks. This is accomplished by utilizing ideas from sequential decision-making and online learning.

There are still a number of restrictions, however. When applied to real-world deployments, the models use the assumption that the agents are rational and that they have complete observability of previous feedback. On top of that, while the simulation-based validations are substantial, it is possible that they do not adequately depict the intricacies of large-scale, heterogeneous EC platforms. In the future, research should investigate hybrid models that include the prediction of user behaviour, assessment of uncertainty, and methods for multi-agent cooperation.

Another disadvantage is the dependence on synthetic datasets and controlled situations, which may not adequately reflect unforeseen occurrences like large-scale failures, security breaches, or user-generated abnormalities. Energy consumption is not explicitly included as an optimisation objective in the current frameworks and is therefore identified as a direction for future work. Future additions may include cross-layer optimisation throughout the network stack, and the incorporation of trust and privacy-preserving methods to enable safe collaborative decision-making among varied stakeholders.

7.3 Future Directions

Looking ahead, the study of node selection in edge computing brings up an extensive number of possible research directions that can considerably improve the flexibility, resilience, and usefulness of bandit-driven algorithms. The fast development of edge computing applications in areas such as autonomous mobility, industrial automation, and immersive services requires decision-making frameworks that are not only accurate but also scalable and resource-efficient. Future research ought to attempt to exceed the limits of present algorithms to function in settings that are bigger in scale, more heterogeneous, and more volatile in behaviour. One preferred path is to incorporate greater types of contextual information into the node selection process. While existing frameworks mostly rely on task-level and system-state feedback, future designs may incorporate multi-modal signals such as user mobility, traffic patterns, and service-level agreements to make more informed judgements. Incorporating such multi-source data requires elaborate fusion techniques and could lead the way for more customised node selection strategies, in which judgements are adjusted to specific application requirements or user desires.

Another important research direction is the incorporation of sustainability and environmental awareness into node selection mechanisms. As the energy cost of edge infrastructures becomes a global concern, algorithms may need to consider energy consumption and environmental impact alongside traditional performance metrics. Integrating such considerations into the incentive or reward structure could influence how nodes are evaluated and selected, encouraging greener computing solutions without compromising system performance. Furthermore, security and trust are becoming critical considerations in distributed edge ecosystems. Future algorithms may include trust scores, anomaly detection, or adversarial resistance in their selection procedures to maintain consistent performance even in adversarial or unreliable settings. This research may tie together multi-armed bandit theory, cybersecurity, and distributed consensus.

A closely related future direction is the systematic analysis of adversarial attacks against bandit-driven node selection mechanisms. In practical EC environments, malicious nodes may attempt to manipulate feedback signals, inject misleading contextual information, or impersonate trustworthy nodes in order to influence selection decisions. Investigating the robustness of the proposed algorithms under such adversarial conditions, and developing attack-resilient variants of the frameworks presented in this thesis, represents an important avenue for future work.

Finally, the practical implementation of these ideas in real-world experiments that range from smart cities to vehicle networks will be a crucial milestone. Such implementations will enable researchers to confirm assumptions, assess real-time scalability, and fine-tune algorithms under actual restrictions. On the basis of the existing study, there are a number of interesting routes that may be investigated:

- **Multi-Agent Coordination:** Extending the models that have been developed to cooperative situations, in which numerous edge nodes coordinate service choices via the use of game-theoretic frameworks or decentralized consensus.
- **Robustness to Uncertainty:** It is possible to manage partial observability and unreliable input data by including probabilistic inference and resilient optimisation methods.
- **Real-World Deployment:** Making the transition from simulation to field testing in smart city or campus-wide testbeds in order to confirm assumptions and increase model generalization.

7.4 Final Remarks

By laying the foundations for robust and intelligent edge computing systems that can operate under uncertainty and resource constraints, this thesis contributes to their development. By combining the concepts of optimal stopping theory, contextual bandits, and expert learning, it makes a contribution to the development of a unified and practical toolkit for making decisions in real time at the edge. Not only do these contributions push the bounds of theoretical possibilities, but they also offer substantial promise for practical implementation in distributed systems of the future generation.

Bibliography

[1] Vinod Veeramachaneni. 'Edge Computing: Architecture, Applications, and Future Challenges in a Decentralized Era'. In: *Recent Trends in Computer Graphics and Multimedia Technology* 7.1 (2025), pp. 8–23.

[2] Hongbo Jiang et al. 'Joint task offloading and resource allocation for energy-constrained mobile edge computing'. In: *IEEE Transactions on Mobile Computing* 22.7 (2022), pp. 4000–4015.

[3] Abdul Rehman Javed et al. 'Future smart cities: Requirements, emerging technologies, applications, challenges, and future aspects'. In: *Cities* 129 (2022), p. 103794.

[4] Saleh AlFahad, Christos Anagnostopoulos and Kostas Kolomvatsos. 'Time-optimized sequential decision making for service management in smart city environments'. In: *Journal of Smart Cities and Society* Preprint (2023), pp. 1–23.

[5] Lina A Haibeh, Mustapha CE Yagoub and Abdallah Jarray. 'A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches'. In: *IEEE Access* 10 (2022), pp. 27591–27610.

[6] Saleh AlFahad et al. 'Task offloading in mobile edge computing using cost-based discounted optimal stopping'. In: *Open Computer Science* 14.1 (2024), p. 20230115.

[7] Shameem A Puthiya Parambath, Christos Anagnostopoulos and Roderick Murray-Smith. 'Sequential query prediction based on multi-armed bandits with ensemble of transformer experts and immediate feedback'. In: *Data Mining and Knowledge Discovery* (2024), pp. 1–25.

[8] Saleh ALFahad et al. 'Node selection using adversarial expert-based multi-armed bandits in distributed computing'. In: *Computing* 107.3 (2025), p. 85.

[9] Mayuko Sato et al. 'Total optimization of energy networks in a smart city by multi-swarm differential evolutionary particle swarm optimization'. In: *IEEE Transactions on Sustainable Energy* 10.4 (2018), pp. 2186–2200.

[10] Mingyang Sun et al. 'Probabilistic peak load estimation in smart cities using smart meter data'. In: *IEEE Transactions on Industrial electronics* 66.2 (2018), pp. 1608–1618.

[11] Ching-Chun Liao, Ting-Sheng Chen and An-Yeu Wu. 'Real-time multi-user detection engine design for IoT applications via modified sparsity adaptive matching pursuit'. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.8 (2019), pp. 2987–3000.

[12] Qi Chen et al. 'A survey on an emerging area: Deep learning for smart city data'. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 3.5 (2019), pp. 392–410.

[13] Xiaolong Xu et al. 'Intelligent offloading for collaborative smart city services in edge computing'. In: *IEEE Internet of Things Journal* 7.9 (2020), pp. 7919–7927.

[14] Kostas Kolomvatsos et al. 'Proactive & time-optimized data synopsis management at the edge'. In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[15] Ibrahim Alghamdi, Christos Anagnostopoulos and Dimitrios P Pezaros. 'On the optimality of task offloading in mobile edge computing environments'. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2019, pp. 1–6.

[16] Zhi Zhou et al. 'Edge intelligence: Paving the last mile of artificial intelligence with edge computing'. In: *Proceedings of the IEEE* 107.8 (2019), pp. 1738–1762.

[17] Olakunle Elijah et al. 'An overview of Internet of Things (IoT) and data analytics in agriculture: Benefits and challenges'. In: *IEEE Internet of things Journal* 5.5 (2018), pp. 3758–3773.

[18] Weisong Shi et al. 'Edge computing: Vision and challenges'. In: *IEEE internet of things journal* 3.5 (2016), pp. 637–646.

[19] Navjeet Kaur et al. 'Healthcare Monitoring Through Fog Computing: A Survey'. In: *ECS Transactions* 107.1 (2022), p. 7689.

[20] Zhe Yu et al. 'Joint task offloading and resource allocation in UAV-enabled mobile edge computing'. In: *IEEE Internet of Things Journal* 7.4 (2020), pp. 3147–3159.

[21] Jianyu Wang et al. 'Edge cloud offloading algorithms: Issues, methods, and perspectives'. In: *ACM Computing Surveys (CSUR)* 52.1 (2019), pp. 1–23.

[22] Jun Zhang et al. 'An architecture for IoT-enabled smart transportation security system: a geospatial approach'. In: *IEEE Internet of Things Journal* 8.8 (2020), pp. 6205–6213.

[23] Saurabh Singh et al. 'SH-BlockCC: A secure and efficient Internet of things smart home architecture based on cloud computing and blockchain technology'. In: *International Journal of Distributed Sensor Networks* 15.4 (2019), p. 1550147719844159.

[24] Malong Ke et al. 'Massive access in cell-free massive MIMO-based Internet of Things: Cloud computing and edge computing paradigms'. In: *IEEE Journal on Selected Areas in Communications* 39.3 (2020), pp. 756–772.

[25] Dmitrii Chemodanov et al. 'AGRA: AI-augmented geographic routing approach for IoT-based incident-supporting applications'. In: *Future Generation Computer Systems* 92 (2019), pp. 1051–1065.

[26] Qinglin Qi and Fei Tao. 'A smart manufacturing service system based on edge computing, fog computing, and cloud computing'. In: *IEEE Access* 7 (2019), pp. 86769–86777.

[27] Najmul Hassan et al. 'The role of edge computing in internet of things'. In: *IEEE communications magazine* 56.11 (2018), pp. 110–115.

[28] Christos Anagnostopoulos, Stathes Hadjiefthymiades and Kostas Kolomvatsos. 'Accurate, dynamic, and distributed localization of phenomena for mobile sensor networks'. In: *ACM Transactions on Sensor Networks (TOSN)* 12.2 (2016), pp. 1–59.

[29] Kostas Kolomvatsos, Christos Anagnostopoulos and Stathes Hadjiefthymiades. 'Data fusion and type-2 fuzzy inference in contextual data stream monitoring'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.8 (2016), pp. 1839–1853.

[30] Kostas Kolomvatsos, Christos Anagnostopoulos and Stathes Hadjiefthymiades. 'An efficient environmental monitoring system adopting data fusion, prediction, & fuzzy logic'. In: *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE. 2015, pp. 1–6.

[31] Kostas Kolomvatsos, Christos Anagnostopoulos and Stathes Hadjiefthymiades. 'Distributed localized contextual event reasoning under uncertainty'. In: *IEEE Internet of Things Journal* 4.1 (2016), pp. 183–191.

[32] Quan Yuan et al. 'A joint service migration and mobility optimization approach for vehicular edge computing'. In: *IEEE Transactions on Vehicular Technology* 69.8 (2020), pp. 9041–9052.

[33] Zhipeng Gao et al. 'Deep reinforcement learning based service migration strategy for edge computing'. In: *2019 IEEE international conference on service-oriented system engineering (SOSE)*. IEEE. 2019, pp. 116–1165.

[34] Qi Gu et al. 'Mobile edge computing via wireless power transfer over multiple fading blocks: An optimal stopping approach'. In: *IEEE Transactions on Vehicular Technology* 69.9 (2020), pp. 10348–10361.

[35] Kostas Kolomvatsos and Christos Anagnostopoulos. 'A proactive statistical model supporting services and tasks management in pervasive applications'. In: *IEEE Transactions on Network and Service Management* 19.3 (2022), pp. 3020–3031.

[36] Farouk Messaoudi, Adlen Ksentini and Philippe Bertin. 'On using edge computing for computation offloading in mobile network'. In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE. 2017, pp. 1–7.

[37] Davide Callegaro and Marco Levorato. 'Optimal edge computing for infrastructure-assisted uav systems'. In: *IEEE Transactions on Vehicular Technology* 70.2 (2021), pp. 1782–1792.

[38] Md Golam Rabiul Alam et al. 'Autonomic computation offloading in mobile edge for IoT applications'. In: *Future Generation Computer Systems* 90 (2019), pp. 149–157.

[39] Thinh Quang Dinh et al. 'Offloading in mobile edge computing: Task allocation and computational frequency scaling'. In: *IEEE Transactions on Communications* 65.8 (2017), pp. 3571–3584.

[40] Thinh Quang Dinh et al. 'Learning for computation offloading in mobile edge computing'. In: *IEEE Transactions on Communications* 66.12 (2018), pp. 6353–6367.

[41] Li Kuang et al. 'Offloading decision methods for multiple users with structured tasks in edge computing for smart cities'. In: *Future Generation Computer Systems* 105 (2020), pp. 717–729.

[42] Feng Wang et al. 'Joint offloading and computing optimization in wireless powered mobile-edge computing systems'. In: *IEEE Transactions on Wireless Communications* 17.3 (2017), pp. 1784–1797.

[43] Min Chen and Yixue Hao. 'Task offloading for mobile edge computing in software defined ultra-dense network'. In: *IEEE Journal on Selected Areas in Communications* 36.3 (2018), pp. 587–597.

[44] Qi Qi et al. 'Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach'. In: *IEEE Transactions on Vehicular Technology* 68.5 (2019), pp. 4192–4203.

[45] Xiaoge Huang et al. 'Energy-efficient offloading decision-making for mobile edge computing in vehicular networks'. In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (2020), pp. 1–16.

[46] Taha Alfakih et al. 'Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA'. In: *IEEE Access* 8 (2020), pp. 54074–54084.

[47] Ming Tang and Vincent WS Wong. 'Deep reinforcement learning for task offloading in mobile edge computing systems'. In: *IEEE Transactions on Mobile Computing* 21.6 (2020), pp. 1985–1997.

[48] Ling Tang and Shibo He. 'Multi-user computation offloading in mobile edge computing: A behavioral perspective'. In: *IEEE Network* 32.1 (2018), pp. 48–53.

[49] Wei Chang et al. 'Offloading decision in edge computing for continuous applications under uncertainty'. In: *IEEE Transactions on Wireless Communications* 19.9 (2020), pp. 6196–6209.

[50] Zhufang Kuang et al. 'Partial offloading scheduling and power allocation for mobile edge computing systems'. In: *IEEE Internet of Things Journal* 6.4 (2019), pp. 6774–6785.

[51] Yanting Wang et al. 'Mobile-edge computing: Partial computation offloading using dynamic voltage scaling'. In: *IEEE Transactions on Communications* 64.10 (2016), pp. 4268–4282.

[52] Ying Chen et al. 'Energy efficient dynamic offloading in mobile edge computing for internet of things'. In: *IEEE Transactions on Cloud Computing* 9.3 (2019), pp. 1050–1060.

[53] Yijin Pan et al. 'Energy-efficient NOMA-based mobile edge computing offloading'. In: *IEEE Communications Letters* 23.2 (2018), pp. 310–313.

[54] Shangguang Wang et al. 'A survey on service migration in mobile edge computing'. In: *IEEE Access* 6 (2018), pp. 23511–23528.

[55] Paolo Bellavista, Alessandro Zanni and Michele Solimando. 'A migration-enhanced edge computing support for mobile devices in hostile environments'. In: *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE. 2017, pp. 957–962.

[56] Md Golam Rabiul Alam, Yan Kyaw Tun and Choong Seon Hong. 'Multi-agent and reinforcement learning based code offloading in mobile fog'. In: *2016 International Conference on Information Networking (ICOIN)*. IEEE. 2016, pp. 285–290.

[57] Shangguang Wang et al. 'Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach'. In: *IEEE Transactions on Mobile Computing* 20.3 (2019), pp. 939–951.

[58] Tao Ouyang, Zhi Zhou and Xu Chen. 'Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing'. In: *IEEE Journal on Selected Areas in Communications* 36.10 (2018), pp. 2333–2345.

[59] Yuxuan Sun, Sheng Zhou and Jie Xu. 'EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks'. In: *IEEE Journal on Selected Areas in Communications* 35.11 (2017), pp. 2637–2646.

[60] Jinliang Xu et al. 'Path selection for seamless service migration in vehicular edge computing'. In: *IEEE Internet of Things Journal* 7.9 (2020), pp. 9040–9049.

[61] Shiqiang Wang et al. 'Dynamic service migration in mobile edge computing based on Markov decision process'. In: *IEEE/ACM Transactions on Networking* 27.3 (2019), pp. 1272–1288.

[62] Tao Ouyang et al. 'Adaptive user-managed service placement for mobile edge computing: An online learning approach'. In: *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE. 2019, pp. 1468–1476.

[63] Yuxuan Sun et al. 'Learning-based task offloading for vehicular cloud computing systems'. In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–7.

[64] Ibrahim Alghamdi, Christos Anagnostopoulos and Dimitrios P Pezaros. 'Data quality-aware task offloading in mobile edge computing: an optimal stopping theory approach'. In: *Future Generation Computer Systems* 117 (2021), pp. 462–479.

[65] Haibo Zhang, Zixin Wang and Kaijian Liu. 'V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks'. In: *China Communications* 17.5 (2020), pp. 266–283.

[66] Xiaolong Xu et al. 'Trust-oriented IoT service placement for smart cities in edge computing'. In: *IEEE Internet of Things Journal* 7.5 (2019), pp. 4084–4091.

[67] Shuyu Chen et al. 'Context-aware online offloading strategy with mobility prediction for mobile edge computing'. In: *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE. 2021, pp. 1–9.

[68] Zezu Liang et al. 'Service Migration for Multi-Cell Mobile Edge Computing'. In: *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE. 2020, pp. 1–6.

[69] Amir Erfan Eshratifar and Massoud Pedram. 'Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment'. In: *Proceedings of the 2018 on Great Lakes Symposium on VLSI*. 2018, pp. 111–116.

[70] Fei Sun et al. 'Cooperative task scheduling for computation offloading in vehicular cloud'. In: *IEEE Transactions on Vehicular Technology* 67.11 (2018), pp. 11049–11061.

[71] Christos Anagnostopoulos and Kostas Kolomvatsos. 'A delay-resilient and quality-aware mechanism over incomplete contextual data streams'. In: *Information Sciences* 355 (2016), pp. 90–109.

[72] Goran Peskir and Albert Shiryaev. *Optimal stopping and free-boundary problems*. Springer, 2006.

[73] Benxia Zheng et al. 'Application of internet of things and edge computing technology in sports tourism services'. In: *Security and Communication Networks* 2021 (2021), pp. 1–10.

[74] Massimo Villari et al. 'Osmosis: The osmotic computing platform for microelements in the cloud, edge, and internet of things'. In: *Computer* 52.8 (2019), pp. 14–26.

[75] N Saranya, K Geetha and C Rajan. 'Data replication in mobile edge computing systems to reduce latency in internet of things'. In: *Wireless Personal Communications* 112 (2020), pp. 2643–2662.

[76] Hisham AlMajed and Ahmad AlMogren. 'A secure and efficient ECC-based scheme for edge computing and internet of things'. In: *Sensors* 20.21 (2020), p. 6158.

[77] Zhengyu Song et al. 'A comprehensive survey on aerial mobile edge computing: Challenges, state-of-the-art, and future directions'. In: *Computer Communications* (2022).

[78] Cheng Zhan et al. 'Multi-UAV-enabled mobile-edge computing for time-constrained IoT applications'. In: *IEEE Internet of Things Journal* 8.20 (2021), pp. 15553–15567.

[79] Kaouther Gasmi et al. 'A survey on computation offloading and service placement in fog computing-based IoT'. In: *The Journal of Supercomputing* 78.2 (2022), pp. 1983–2014.

[80] Miao Li et al. 'Priority-mece: a mobile edge cloud ecosystem based on priority tasks offloading'. In: *Mobile Networks and Applications* 27.4 (2022), pp. 1768–1777.

[81] Syed Luqman Shah et al. 'TAMEC: trusted augmented mobile execution on cloud'. In: *Scientific Programming* 2021 (2021), pp. 1–8.

[82] Mohammed Maray and Junaid Shuja. 'Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues'. In: *Mobile Information Systems* 2022 (2022).

[83] Madalena Soula et al. 'Intelligent tasks allocation at the edge based on machine learning and bio-inspired algorithms'. In: *Evolving Systems* 13.2 (2022), pp. 221–242.

[84] Pavel V Gapeev and Libo Li. 'Optimal stopping problems for maxima and minima in models with asymmetric information'. In: *Stochastics* 94.4 (2022), pp. 602–628.

[85] Ying Chen et al. 'Dynamic task offloading for mobile edge computing with hybrid energy supply'. In: *Tsinghua Science and Technology* 28.3 (2022), pp. 421–432.

[86] Rui Zhang et al. 'Task offloading with task classification and offloading nodes selection for MEC-Enabled IoV'. In: *ACM Transactions on Internet Technology (TOIT)* 22.2 (2021), pp. 1–24.

[87] Jiagang Liu et al. 'Efficient dependent task offloading for multiple applications in MEC-cloud system'. In: *IEEE Transactions on Mobile Computing* (2021).

[88] Feng Wang, Jie Xu and Shuguang Cui. 'Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems'. In: *IEEE Transactions on Wireless Communications* 19.4 (2020), pp. 2443–2459.

[89] Haifeng Lu et al. 'Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning'. In: *Future Generation Computer Systems* 102 (2020), pp. 847–861.

[90] Pham Van Khanh. 'Optimal stopping time to buy an asset when growth rate is a two-state markov chain'. In: *American Journal of Operations Research* 2014 (2014).

[91] Ivana Nikoloska and Nikola Zlatanov. 'Data selection scheme for energy efficient supervised learning at IoT nodes'. In: *IEEE Communications Letters* 25.3 (2020), pp. 859–863.

[92] Yiping Kang et al. 'Neurosurgeon: Collaborative intelligence between the cloud and mobile edge'. In: *ACM SIGARCH Computer Architecture News* 45.1 (2017), pp. 615–629.

[93] Surat Teerapittayanon, Bradley McDanel and Hsiang-Tsung Kung. 'Distributed deep neural networks over the cloud, the edge and end devices'. In: *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE. 2017, pp. 328–339.

[94] Xing Chen et al. 'Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments'. In: *IEEE Transactions on Parallel and Distributed Systems* 33.3 (2021), pp. 683–697.

[95] Shuran Sheng et al. 'Deep reinforcement learning-based task scheduling in iot edge computing'. In: *Sensors* 21.5 (2021), p. 1666.

[96] Jin Wang et al. 'Fast adaptive task offloading in edge computing based on meta reinforcement learning'. In: *IEEE Transactions on Parallel and Distributed Systems* 32.1 (2020), pp. 242–253.

[97] Zeinab Zabihi, Amir Masoud Eftekhari Moghadam and Mohammad Hossein Rezvani. 'Reinforcement learning methods for computation offloading: a systematic review'. In: *ACM Computing Surveys* 56.1 (2023), pp. 1–41.

[98] Xiong Wang, Jiancheng Ye and John CS Lui. 'Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach'. In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE. 2022, pp. 1199–1208.

[99] Junge Zhu et al. 'Multi-interface channel allocation in fog computing systems using thompson sampling'. In: *IEEE Internet of Things Journal* 8.17 (2021), pp. 13542–13554.

[100] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[101] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. USA: Cambridge University Press, 2006. ISBN: 0521841089.

[102] Lihong Li et al. 'A contextual-bandit approach to personalized news article recommendation'. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.

[103] Deeksha Sinha. 'Optimization for online platforms'. PhD thesis. Massachusetts Institute of Technology, 2021.

[104] Cagatay Sonmez et al. 'Machine learning-based workload orchestrator for vehicular edge computing'. In: *IEEE Transactions on Intelligent Transportation Systems* 22.4 (2020), pp. 2239–2251.

[105] Rui Zhao et al. 'Deep reinforcement learning based mobile edge computing for intelligent Internet of Things'. In: *Physical Communication* 43 (2020), p. 101184.

[106] Yufeng Zhan et al. 'A deep reinforcement learning based offloading game in edge computing'. In: *IEEE Transactions on Computers* 69.6 (2020), pp. 883–893.

[107] Tong Liu et al. 'Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing'. In: *IEEE Transactions on Mobile Computing* 22.7 (2022), pp. 3870–3881.

[108] Huizi Xiao et al. 'Resource optimization of mab-based reputation management for data trading in vehicular edge computing'. In: *IEEE Transactions on Wireless Communications* (2023).

[109] Marcos Kawazoe Aguilera, Wei Chen and Sam Toueg. 'Heartbeat: A timeout-free failure detector for quiescent reliable communication'. In: *Distributed Algorithms: 11th International Workshop, WDAG'97 Saarbrücken, Germany, September 24–26, 1997 Proceedings 11*. Springer. 1997, pp. 126–140.

[110] Bram van Berlo, Aaqib Saeed and Tanir Ozcelebi. 'Towards federated unsupervised representation learning'. In: *Proceedings of the third ACM international workshop on edge systems, analytics and networking*. 2020, pp. 31–36.

[111] Shameem Puthiya Parambath, Christos Anagnostopoulos and Saleh Abdullah M Alfahad. 'Thompson sampling-based recursive block elimination for dynamic assignment under limited budget in pure-exploration'. In: *Data Mining and Knowledge Discovery* 39.1 (2025), p. 7.

[112] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[113] Peter Auer et al. 'The nonstochastic multiarmed bandit problem'. In: *SIAM journal on computing* 32.1 (2002), pp. 48–77.

[114] Yevgeny Seldin et al. 'Evaluation and analysis of the performance of the EXP3 algorithm in stochastic environments'. In: *European Workshop on Reinforcement Learning*. PMLR. 2013, pp. 103–116.

[115] Shuzhen Chen et al. 'Distributed learning dynamics of multi-armed bandits for edge intelligence'. In: *Journal of Systems Architecture* 114 (2021), p. 101919.

[116] Chien-Sheng Yang, Ramtin Pedarsani and A Salman Avestimehr. 'Edge computing in the dark: Leveraging contextual-combinatorial bandit and coded computing'. In: *IEEE/ACM Transactions on Networking* 29.3 (2021), pp. 1022–1031.

[117] Josh Broch et al. 'A performance comparison of multi-hop wireless ad hoc network routing protocols'. In: *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. 1998, pp. 85–97.

[118] Teng Fei et al. 'How to select a good alternate path in large peer-to-peer systems?' In: *INFOCOM*. 2006.

[119] Punam Bedi and Chhavi Sharma. 'Community detection in social networks'. In: *Wiley interdisciplinary reviews: Data mining and knowledge discovery* 6.3 (2016), pp. 115–135.

[120] Flávia Delicato et al. 'An efficient heuristic for selecting active nodes in wireless sensor networks'. In: *Computer Networks* 50.18 (2006), pp. 3701–3720.

[121] Haining Chen, Hongyi Wu and N-F Tzeng. 'Grid-based approach for working node selection in wireless sensor networks'. In: *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*. Vol. 6. IEEE. 2004, pp. 3673–3678.

[122] Xiaofei Wang et al. 'Convergence of edge computing and deep learning: A comprehensive survey'. In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 869–904.

[123] Yun Zhou, Yuguang Fang and Yanchao Zhang. 'Securing wireless sensor networks: a survey'. In: *IEEE Communications Surveys & Tutorials* 10.3 (2008), pp. 6–28.

[124] Jinlai Xu, Balaji Palanisamy and Qingyang Wang. 'Resilient stream processing in edge computing'. In: *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. 2021, pp. 504–513.

[125] Shameem A Puthiya Parambath et al. 'Sequential Block Elimination for Dynamic Pricing'. In: (2024).

[126] Yuval Lewi, Haim Kaplan and Yishay Mansour. 'Thompson sampling for adversarial bit prediction'. In: *Algorithmic Learning Theory*. PMLR. 2020, pp. 518–553.

[127] Jincheng Mei et al. 'Stochastic gradient succeeds for bandits'. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 24325–24360.

[128] Amélie Héliou, Panayotis Mertikopoulos and Zhengyuan Zhou. 'Gradient-free online learning in continuous games with delayed rewards'. In: *International conference on machine learning*. PMLR. 2020, pp. 4172–4181.

[129] Nicolas Gutowski et al. 'Gorthaur-EXP3: Bandit-based selection from a portfolio of recommendation algorithms balancing the accuracy-diversity dilemma'. In: *Information Sciences* 546 (2021), pp. 378–396.

[130] Varsha Dani, Sham M Kakade and Thomas Hayes. 'The price of bandit information for online optimization'. In: *Advances in Neural Information Processing Systems* 20 (2007).

[131] Yoav Freund and Robert E Schapire. 'A decision-theoretic generalization of on-line learning and an application to boosting'. In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[132] Zhida Jiang et al. 'Fedmp: Federated learning through adaptive model pruning in heterogeneous edge computing'. In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE. 2022, pp. 767–779.

[133] Kevin Jamieson and Ameet Talwalkar. 'Non-stochastic best arm identification and hyperparameter optimization'. In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 240–248.

[134] Kanishka Misra, Eric M Schwartz and Jacob Abernethy. 'Dynamic online pricing with incomplete information using multiarmed bandit experiments'. In: *Marketing Science* 38.2 (2019), pp. 226–252.

[135] Jonas W Mueller, Vasilis Syrgkanis and Matt Taddy. 'Low-rank bandit methods for high-dimensional dynamic pricing'. In: *Advances in Neural Information Processing Systems* 32 (2019).

[136] Yiyun Luo, Will Wei Sun et al. 'Distribution-free contextual dynamic pricing'. In: *arXiv preprint arXiv:2109.07340* (2021).

[137] Marco Mussi et al. 'Pricing the long tail by explainable product aggregation and monotonic bandits'. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 3623–3633.

[138] H. Brendan McMahan and Matthew Streeter. 'Tighter Bounds for Multi-Armed Bandits with Expert Advice'. In: *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*. 2009.

[139] Yuriy Gorodnichenko, Viacheslav Sheremirov and Oleksandr Talavera. 'Price setting in online markets: Does IT click?' In: *Journal of the European Economic Association* 16.6 (2018), pp. 1764–1811.

[140] G Tomas M Hult et al. 'Antecedents and consequences of customer satisfaction: do they differ across online and offline purchases?' In: *Journal of Retailing* 95.1 (2019), pp. 10–23.

[141] Els Breugelmans and Katia Campo. 'Cross-channel effects of price promotions: An empirical analysis of the multi-channel grocery retail sector'. In: *Journal of Retailing* 92.3 (2016), pp. 333–351.

[142] Sébastien Bubeck et al. 'X-Armed Bandits.' In: *Journal of Machine Learning Research* 12.5 (2011).

[143] Sébastien Bubeck, Rémi Munos and Gilles Stoltz. 'Pure exploration in finitely-armed and continuous-armed bandits'. In: *Theoretical Computer Science* 412.19 (2011), pp. 1832–1852.

[144] Jean-Yves Audibert and Sébastien Bubeck. 'Best Arm Identification in Multi-Armed Bandits'. In: *COLT-23th Conference on Learning Theory-2010*. 2010, 13–p.

[145] Victor Gabillon, Mohammad Ghavamzadeh and Alessandro Lazaric. 'Best arm identification: A unified approach to fixed budget and fixed confidence'. In: *Advances in Neural Information Processing Systems* 25 (2012).

[146] Shahin Shahrampour, Mohammad Noshad and Vahid Tarokh. 'On sequential elimination algorithms for best-arm identification in multi-armed bandits'. In: *IEEE Transactions on Signal Processing* 65.16 (2017), pp. 4281–4292.

[147] Honglei Zhuang, Chi Wang and Yifan Wang. 'Identifying outlier arms in multi-armed bandit'. In: *Advances in Neural Information Processing Systems* 30 (2017).

[148] Yikun Ban and Jingrui He. 'Generic outlier detection in multi-armed bandit'. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 913–923.

[149] Yinglun Zhu, Sumeet Katariya and Robert Nowak. 'Robust outlier arm identification'. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11566–11575.

[150] Stefan Magureanu, Richard Combes and Alexandre Proutiere. 'Lipschitz bandits: Regret lower bound and optimal algorithms'. In: *Conference on Learning Theory*. PMLR. 2014, pp. 975–999.

[151] Chara Podimata and Alex Slivkins. 'Adaptive discretization for adversarial Lipschitz bandits'. In: *Conference on Learning Theory*. PMLR. 2021, pp. 3788–3805.

[152] Peter Auer, Nicolo Cesa-Bianchi and Paul Fischer. 'Finite-time analysis of the multiarmed bandit problem'. In: *Machine learning* 47.2 (2002), pp. 235–256.

[153] Zohar Karnin, Tomer Koren and Oren Somekh. 'Almost optimal exploration in multi-armed bandits'. In: *International Conference on Machine Learning*. PMLR. 2013, pp. 1238–1246.

[154] Andrea Tirinzoni and Rémy Degenne. 'On Elimination Strategies for Bandit Fixed-Confidence Identification'. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 18586–18598.

[155] Olivier Chapelle and Lihong Li. 'An empirical evaluation of thompson sampling'. In: *Advances in neural information processing systems* 24 (2011).

[156] Shipra Agrawal and Navin Goyal. 'Analysis of thompson sampling for the multi-armed bandit problem'. In: *Conference on learning theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 39–1.

[157] Shipra Agrawal and Navin Goyal. 'Thompson sampling for contextual bandits with linear payoffs'. In: *International conference on machine learning*. PMLR. 2013, pp. 127–135.

[158] Daniel Russo. 'Simple bayesian algorithms for best arm identification'. In: *Conference on Learning Theory*. PMLR. 2016, pp. 1417–1418.

[159] Marc Jourdan et al. 'Top two algorithms revisited'. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 26791–26803.

[160] Siwei Wang and Jun Zhu. 'Thompson sampling for (combinatorial) pure exploration'. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23470–23483.

[161] Robert Kleinberg, Aleksandrs Slivkins and Eli Upfal. 'Multi-armed bandits in metric spaces'. In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 2008, pp. 681–690.

[162] Robert Kleinberg, Aleksandrs Slivkins and Eli Upfal. 'Bandits and experts in metric spaces'. In: *Journal of the ACM (JACM)* 66.4 (2019), pp. 1–77.

[163] Alexandra Carpentier and Andrea Locatelli. 'Tight (lower) bounds for the fixed budget best arm identification bandit problem'. In: *Conference on Learning Theory*. PMLR. 2016, pp. 590–604.

[164] Marco Faella, Alberto Finzi and Luigi Sauro. 'Rapidly finding the best arm using variance'. In: *ECAI 2020*. IOS Press, 2020, pp. 2585–2591.

[165] Jianyu Xu and Yu-Xiang Wang. 'Towards agnostic feature-based dynamic pricing: Linear policies vs linear valuation with unknown noise'. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 9643–9662.

[166] Yiyun Luo, Will Wei Sun and Yufeng Liu. 'Contextual Dynamic Pricing with Unknown Noise: Explore-then-UCB Strategy and Improved Regrets'. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 37445–37457.

[167] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[168] Pierre-Arnaud Coquelin and Rémi Munos. 'Bandit Algorithms for Tree Search'. In: *Uncertainty in Artificial Intelligence*. 2007.

[169] Sanjay Chawla and Aristides Gionis. 'k-means–: A unified approach to clustering and outlier detection'. In: *Proceedings of the 2013 SIAM international conference on data mining*. SIAM. 2013, pp. 189–197.

[170] *Closed-Loop Data Science Interaction with Probabilistic Models*. https://www.gla.ac.uk/schools/computing/research/researchsections/ida-section/closedloop/. Accessed: 2023-11-29.

[171] Roderick Murray-Smith Sebastian Stein. *Interactive Model-based Probabilistic Visualisations for Exploring Decisions*. Tech. rep. University of Glasgow, School Of Compting Science, July 2023.

[172] Brendan McMahan et al. 'Communication-efficient learning of deep networks from decentralized data'. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[173] Qianyu Long et al. 'Feddip: Federated learning with extreme dynamic pruning and incremental regularization'. In: *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2023, pp. 1187–1192.