A Hierarchical Incentive Mechanism Design for Clustered Federated Reinforcement Learning with Budget Limitation: A Contract-Stackelberg Game Framework

Shaghayegh Ghasemi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (MASc) at

Concordia University

Montréal, Québec, Canada

May 2025

Abstract

A Hierarchical Incentive Mechanism Design for Clustered Federated Reinforcement Learning with Budget Limitation: A Contract-Stackelberg Game Framework

Shaghayegh Ghasemi

Federated Reinforcement Learning (FRL) offers a powerful framework for distributed autonomous agents to collaboratively learn decision-making policies while preserving data privacy. This is particularly valuable in domains such as connected autonomous vehicles, where data sensitivity and environmental variability present major challenges. However, practical FRL systems face several obstacles, including environmental heterogeneity, high participation costs, and limited budget availability at the server side for incentivizing agents. To address these issues, we propose a clustered FRL framework that organizes agents into groups based on their operational environments, enabling more efficient training and localized coordination. Each cluster is managed by a local server that aggregates agent updates and forwards refined models to a main server for global aggregation. To encourage sustained participation, we design a hierarchical incentive mechanism: at the lower layer, contract theory is employed due to information asymmetry; at the upper layer, a Stackelberg game is formulated to enable the main server to allocate its limited budget strategically across clusters based on the accuracy of the local servers' trained models.

Keywords: Federated Learning, Reinforcement Learning, Autonomous Driving, Incentive Design, Stackelberg Game, Contract Theory, Budget-Constrained Optimization

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Jun Cai, for his invaluable guidance, unwavering support, and insightful feedback throughout my research journey. His expertise and mentorship have been instrumental in shaping this work, and I am truly fortunate to have had the opportunity to learn under his supervision.

I am also immensely grateful for the financial support provided by Dr. Jun Cai, which enabled me to focus on my research without financial constraints. His generosity and commitment to fostering academic growth have made this work possible.

I extend my heartfelt appreciation to my colleagues and collaborators at Concordia University for their stimulating discussions, constructive feedback, and encouragement. Their insights have significantly contributed to the development of this research.

A special thanks to my family and friends for their unwavering support, patience, and encouragement throughout this journey. Their belief in my abilities has been a constant source of motivation.

Finally, I acknowledge the resources and support provided by Concordia University, which have facilitated my research.

Contents

List of Figures viii

List of Tables ix

1 Introduction 1

1.1 Background and Motivation . . . 1

1.2 Research Problem and Objectives . . . 3

1.3 Proposed Approach and Contributions . . . 4

1.4 Thesis Organization . . . 5

2 Literature Review 6

2.1 Reinforcement Learning . . . 7

2.1.1 Historical Foundations and Core Concepts . . . 7

2.1.2 Advancements in Reinforcement Learning Algorithms . . . 7

2.1.3 Applications of Reinforcement Learning . . . 8

2.1.4 Challenges and Limitations of Reinforcement Learning . . . 9

2.2 Federated Learning . . . 9

2.2.1 Types of Federated Learning . . . 10

2.2.2 Federated Learning in Real-World Applications . . . 11

2.2.3 Challenges in Federated Learning . . . 12

2.3 Hierarchical and Clustered Federated Learning . . . 13

2.3.1 Hierarchical Federated Learning . . . 13

2.3.2 Clustered Federated Learning 14
2.4 Federated Reinforcement Learning 14
2.4.1 Federated Reinforcement Learning Architectures 15
2.4.2 Recent Advances in FRL 16
2.4.3 Applications of FRL 17
2.4.4 Challenges and Open Issues 18
2.5 Incentive Mechanisms 19
2.5.1 Incentive Mechanisms in FL 20
2.5.2 Contract Theory-Based Incentives 21
2.5.3 Stackelberg Game-Based Incentives 23
2.5.4 Comparison and Discussion 24

3 System Model and Problem Formulation **27**
3.1 System Model 27
3.2 Problem Formulation 32
3.2.1 Budget Constraint 32
3.2.2 Contract Design 34
3.2.3 Stackelberg Game Framework 35

4 Contract-Stackelberg Game Framework **38**
4.1 Dynamic Contract Design 38
4.2 Stackelberg Game Design 44

5 Simulation Results **51**
5.1 Experiment Setup 51
5.2 Experiment Results 53

6 Conclusions and Future Work **61**
6.1 Conclusions 61
6.2 Future Work 62

Appendix A Appendix **65**

Bibliography **68**

List of Figures

Figure 3.1 Clustered federated reinforcement learning framework with a two-layer hierarchical structure. 28

Figure 4.1 Logistic curve illustrating the relationship between total accuracy and budget data points for each cluster. 45

Figure 5.1 Utility of local servers as a function of budget allocation. 53

Figure 5.2 Data contribution vs. budget for Clusters 1 and 2. 54

Figure 5.3 Data contribution vs. budget for Clusters 3 and 4. 55

Figure 5.4 IR and IC constraint satisfaction for Clusters 1 and 2 in the last round of training. 56

Figure 5.5 IR and IC constraint satisfaction for Clusters 3 and 4 in the last round of training. 57

Figure 5.6 Main server utility as a function of total budget T. 59

Figure 5.7 Comparison of main server utility and total budget usage across models . . 60

List of Tables

Table 3.1 Important Notations 30

Table 5.1 Evaluation Parameters 52

Table 5.2 Comparison of $q_{\max}$, q, Reward, and θ for the Last Round Across Clusters . . 58

Chapter 1

Introduction

1.1 Background and Motivation

As intelligent agents become increasingly autonomous and geographically distributed, the demand for collaborative learning frameworks that preserve privacy, operate under resource constraints, and adapt to diverse environments has grown substantially. Traditional centralized machine learning approaches, reliant on aggregating data at a central server, are increasingly infeasible due to privacy regulations, bandwidth limitations, and the computational constraints of edge devices [1,2].

Federated Learning (FL) [1] offers a decentralized alternative. In FL, agents train local models on private data and transmit only model updates (e.g., gradients or parameters) to a central server. This preserves data privacy and reduces communication overhead, making FL particularly useful in mobile computing, healthcare, and personalized services [2]. However, FL is primarily designed for supervised learning tasks using fixed datasets and lacks mechanisms for handling sequential decision-making or interactive learning.

Reinforcement Learning (RL), by contrast, is suited for dynamic environments in which agents learn optimal policies through sequential interactions with their surroundings. Integrating FL with RL gives rise to Federated Reinforcement Learning (FRL) [3,4], which enables decentralized agents to learn from their local environments while contributing to a global policy. FRL holds great promise in domains such as autonomous driving, industrial IoT, smart grids, and multi-robot systems—contexts where environmental feedback is local and privacy is crucial [5].

However, deploying FRL in real-world environments presents several non-trivial challenges. A central issue is heterogeneity, which arises in multiple forms: agents may differ in their environmental dynamics, task structures, computational resources, and communication abilities [6]. These disparities can significantly impact both local learning effectiveness and the quality of aggregated global updates, leading to slower convergence or model collapse [7, 8]. Furthermore, participating in collaborative training often incurs non-negligible costs, such as local computation, energy usage, and network bandwidth [9], making it rational for agents to disengage unless properly incentivized.

A key distinction between FRL and traditional FL lies in the nature of data generation. While FL typically assumes fixed datasets and consistent model updates across rounds, FRL agents continuously interact with their environments, generating an evolving and non-stationary stream of experience data. This dynamic nature introduces additional complexity in synchronization, policy evaluation, and cross-agent knowledge transfer [10, 11]. As a result, incentive mechanisms originally developed for FL—which assume static data and periodic participation—may prove inadequate in FRL settings. Recent efforts have extended contract-based incentive models to FRL by accounting for data accumulation across rounds [12], yet these approaches largely overlook hierarchical coordination and budget limitations.

To tackle such heterogeneity and enable scalable deployment, FRL systems increasingly adopt hierarchical architectures. In these systems, agents are clustered according to contextual similarities, such as geographical proximity or task relevance, and each cluster is managed by a local server. These local servers perform intra-cluster coordination and periodically communicate with a main server, which integrates knowledge across clusters [13, 14]. This structure facilitates both local specialization and global generalization. In scenarios like vehicular networks, where traffic patterns and environmental conditions differ across regions, hierarchical FRL enables context-aware policy learning while maintaining global consistency [15].

We adopt such a clustered FRL framework as the underlying structure for our study. While this setup has proven effective in managing heterogeneity, it also introduces a complex challenge that is underexplored in the current literature: how to design effective incentive mechanisms under tight budget constraints and multi-layered coordination. In these settings, the main server must decide how to allocate a budget to each local server, and each local server must then design its own strategy to incentivize agent contributions within its cluster. These decisions are tightly coupled: inefficiencies at one layer can propagate and degrade performance across the entire hierarchy.

Despite increasing attention to incentive mechanisms in FL [16, 17], most works treat budget constraints at a single layer and do not model how hierarchical coordination structures complicate budget allocation. Additionally, few studies consider how to balance long-term engagement, increasing data volumes, and strategic agent behavior under asymmetric information. These gaps limit the applicability of existing methods in realistic FRL settings.

Taken together, these challenges—environmental heterogeneity, dynamic data accumulation, strategic agent behavior, information asymmetry, and limited budgets—underscore the need for a principled and scalable incentive framework tailored to FRL's unique demands. The following section introduces the specific research problem addressed in this thesis and outlines the proposed solution.

1.2 Research Problem and Objectives

This thesis addresses the design of a hierarchical, incentive-compatible mechanism for FRL systems operating under strict budget constraints and incomplete information. The central objective is to develop a scalable framework that ensures reliable agent participation while maximizing overall system utility in environments characterized by layered coordination and asymmetric knowledge of agent capabilities.

To this end, we consider a clustered FRL architecture in which agents are grouped based on contextual factors such as location, task similarity, or environmental conditions. Each cluster is managed by a local server responsible for coordinating intra-cluster learning, while a central main server oversees global policy integration. This structure reflects real-world applications such as autonomous transportation systems, distributed robotics, and mobile edge computing.

Within this architecture, two key challenges emerge. First, local servers must incentivize participation without full knowledge of agent characteristics, such as computational capacity or energy cost. This information asymmetry complicates fair and efficient contract design, as agents may behave strategically to maximize their own benefit while minimizing contribution. Second, budget allocation decisions are inherently interdependent across layers: insufficient funding from the main server can impair local incentive schemes, resulting in degraded learning outcomes that, in turn, influence future global allocations. These challenges necessitate a coordinated incentive mechanism that operates coherently across system layers.

We address these issues through a hierarchical incentive model implemented at two levels. At the lower layer, contract theory is employed to design reward mechanisms that guarantee incentive compatibility (IC) and individual rationality (IR) under asymmetric information. At the upper layer, a Stackelberg game framework enables the main server (leader) to strategically allocate budget across clusters by anticipating the local servers' (followers') responses in terms of utility optimization. This combination ensures that incentives are both fair and aligned with system-wide learning objectives, enabling cost-effective deployment in budget-limited FRL settings.

1.3 Proposed Approach and Contributions

This thesis introduces a novel hierarchical incentive mechanism that integrates economic theory into FRL systems with budget limitations. Our framework addresses the dual challenges of asymmetric information and budget coordination through a unified optimization model. The key contributions are as follows:

(1) We design a hierarchical incentive mechanism that integrates contract theory at the lower layer and Stackelberg game theory at the upper layer for strategic budget allocation by the main server.

(2) We introduce a budget-centric perspective to FRL incentive design, shifting the focus from agent-side resource constraints to main server-oriented budget optimization, which better reflects real-world deployment considerations.

(3) We introduce a backward induction algorithm in the hierarchical structure of mechanism design, investigating the incentive coupling between the two layers.

(4) We provide both theoretical analysis and empirical validation of the proposed framework, demonstrating that it maximizes the main server's utility, reduces overall incentive costs,

and ensures agent engagement by guaranteeing Incentive Compatibility (IC) and Individual Rationality (IR).

Our work has been submitted to IEEE Transactions on Mobile Computing.

1.4 Thesis Organization

The remainder of this thesis is organized into five chapters. Chapter 2 provides a comprehensive review of relevant literature in FL, RL, FRL, and incentive mechanisms, highlighting key challenges and research gaps that motivate this study. Chapter 3 introduces the system model and formally defines the problem setting, including the assumptions and constraints considered. Chapter 4 details the proposed incentive framework, including the contract design process and global strategy derivation using Stackelberg game theory. Chapter 5 presents simulation experiments and performance evaluations, demonstrating the effectiveness of the proposed mechanism in clustered FRL settings. Finally, Chapter 6 concludes the thesis with a summary of key findings, a discussion of limitations, and potential directions for future work.

Chapter 2

Literature Review

The advancement of autonomous and distributed systems has led to a growing demand for intelligent decision-making frameworks that can operate efficiently while preserving data privacy. In this context, reinforcement learning has emerged as a powerful approach for training autonomous agents through interaction with dynamic environments. Simultaneously, federated learning has gained significant attention as a decentralized learning paradigm that enables multiple agents to collaboratively train machine learning models without directly sharing their local data.

The integration of these two methodologies, known as Federated Reinforcement Learning (FRL), has shown promising potential in various real-world applications, including autonomous driving, smart grids, and edge computing. However, several challenges arise in FRL, particularly in handling heterogeneous environments, addressing resource constraints, and developing scalable and effective incentive mechanisms to ensure sustained agent participation.

This chapter surveys the foundational principles of reinforcement learning and federated learning, explores their integration into FRL, and examines existing work on incentive mechanisms, focusing on contract theory and Stackelberg game models, as a basis for the proposed contributions in this thesis. It also reviews hierarchical and clustered extensions of federated learning that address scalability and heterogeneity issues. By critically examining the current landscape and its limitations, this review provides the necessary foundation for exploring novel incentive mechanisms that are better suited to the complexities of FRL.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where agents learn to make decisions by interacting with an environment to maximize cumulative rewards. Unlike supervised learning, which relies on labeled datasets, RL agents learn optimal behaviors through trial-and-error experiences, receiving feedback in the form of rewards or penalties. This paradigm is particularly suited for problems where decision-making sequences significantly impact outcomes, such as robotics, game playing, and autonomous driving [18].

2.1.1 Historical Foundations and Core Concepts

The conceptual underpinnings of RL can be traced back to early studies in animal learning and behavioral psychology, where behaviors were understood to be shaped by rewards and punishments [19]. This perspective is rooted in the principles of operant conditioning introduced by B.F. Skinner laid the foundation for modern RL algorithms. In the realm of computer science, these ideas have been formalized using the framework of Markov Decision Processes (MDPs), which provide a mathematical model for decision-making situations where outcomes are partly random and partly under the control of a decision-maker [20].

An MDP consists of a set of states, a set of actions, transition probabilities between states, and reward functions. The objective of an MDP is to determine a policy, a mapping from states to actions, that maximizes the expected cumulative reward over time [19]. One of the seminal works in this area is by Kaelbling et al., who provided a comprehensive survey of RL methods, discussing central issues such as the exploration-exploitation trade-off, delayed rewards, and generalization in RL systems. They highlighted the challenge of balancing exploration (discovering new actions for long-term rewards) and exploitation (choosing known actions that yield immediate high rewards) [18].

2.1.2 Advancements in Reinforcement Learning Algorithms

Over the years, RL has evolved significantly, with advancements in algorithms enhancing its applicability and efficiency. Traditional methods like Dynamic Programming require a complete model of the environment's dynamics, which is often impractical. Model-free methods, such as Monte Carlo and Temporal Difference (TD) learning, have been developed to learn optimal policies without explicit environmental models. Q-learning, a form of TD learning introduced by Watkins, enables agents to learn the value of actions in specific states, facilitating the derivation of optimal policies without requiring a model of state transition probabilities [21].

The integration of deep learning techniques with RL has led to the emergence of Deep Reinforcement Learning (DRL), which combines the representation learning capabilities of deep neural networks with the decision-making framework of RL. This integration has enabled significant breakthroughs in complex tasks with high-dimensional state spaces. For instance, DRL has been successfully applied in playing Atari games directly from raw pixel inputs, demonstrating the potential of these methods in handling complex sensory data [22]. However, despite its successes, instabilities and sample inefficiencies remain critical challenges in DRL training [23].

2.1.3 Applications of Reinforcement Learning

RL has been successfully applied in diverse domains, demonstrating its ability to solve complex decision-making problems. In robotics, RL enables autonomous agents to learn optimal control strategies for tasks such as grasping, navigation, and manipulation [24]. Similarly, in healthcare, RL is used to optimize treatment plans, drug discovery, and personalized medicine by modeling sequential decision-making processes [25].

In finance, RL-based trading algorithms help in portfolio optimization and market-making strategies by adapting to dynamic market conditions [24]. RL has also revolutionized gaming and artificial intelligence, with deep RL models achieving superhuman performance in games like Go, chess, and video games [25]. Furthermore, in autonomous driving, RL aids in learning safe and efficient driving policies through simulation-based training.

Other notable applications include industrial automation [26], energy management [27], and smart grid optimization [28], where RL contributes to improving efficiency and resource allocation. Despite its promising applications, challenges such as safety, scalability, and real-world generalization continue to shape RL's practical adoption.

2.1.4 Challenges and Limitations of Reinforcement Learning

Despite its successes, RL faces several challenges that limit its practical adoption. A primary concern is its high sample complexity, as training often requires extensive interactions with the environment, which can be both time-consuming and computationally expensive [29]. While model-based RL and transfer learning techniques have been introduced to improve efficiency, generalization to unseen scenarios remains a significant obstacle [30].

Instability during training is another major limitation. RL algorithms are often sensitive to hyperparameters and prone to issues like catastrophic forgetting, especially when integrated with deep neural networks. These factors can lead to unpredictable performance and complicate deployment in real-world systems [31]. Safety concerns further hinder RL adoption in high-stakes applications, where trial-and-error learning can result in harmful decisions. Ensuring safe exploration and minimizing the risk of undesirable behaviors remains an open challenge [29, 30].

Additionally, RL models typically lack transparency and interpretability, making it difficult to understand or explain their decision-making processes [31]. This black-box nature reduces trust and hinders acceptance in domains requiring accountability or regulatory compliance.

Recent efforts have extended RL to distributed settings, such as FRL, which allows multiple agents to collaboratively learn policies while preserving data privacy. Although FRL addresses key issues related to decentralization and data sharing, it inherits and amplifies many of RL's challenges—particularly in terms of system heterogeneity, communication efficiency, and scalability. These concerns are further explored in the following sections.

2.2 Federated Learning

Federated Learning (FL) is a decentralized machine learning paradigm that enables collaborative model training across multiple clients while keeping data localized, addressing privacy and communication efficiency concerns. Unlike traditional centralized learning, where data is collected and processed on a central server, FL allows distributed edge devices or organizations to train models locally and share only model updates with an aggregator, reducing the risk of data exposure [32].

A key component of FL is model aggregation, which ensures that local model updates from different clients are effectively combined to produce a globally optimized model. The most widely used aggregation method is Federated Averaging (FedAvg), which efficiently merges local updates while minimizing communication costs [1]. Other approaches, such as FedProx, improve model stability by addressing statistical heterogeneity across clients [6]. Secure aggregation techniques, including homomorphic encryption and secure multi-party computation, further enhance privacy by ensuring that only aggregated updates are revealed without exposing individual client data [33]. These aggregation strategies are fundamental to enabling scalable and privacy-preserving federated learning systems.

2.2.1 Types of Federated Learning

Federated Learning can be categorized into different types based on data distribution and relationships among participating clients. The primary types include horizontal federated learning, vertical federated learning, and federated transfer learning, each suited for specific data distribution scenarios [34].

Horizontal federated learning, also known as sample-based FL, is applied when different clients share the same feature space but have different user samples. This is common in scenarios where multiple organizations, such as hospitals or financial institutions, have data with similar attributes but collected from different users [34]. Horizontal federated learning enables them to collaboratively train models while preserving data privacy.

Vertical federated learning, or feature-based FL, occurs when different organizations have datasets containing overlapping users but with different feature sets. For example, a bank and an e-commerce platform may have data on the same customers but with distinct attributes, such as financial history and purchasing behavior. Vertical federated learning allows these entities to jointly train models while maintaining confidentiality by leveraging encryption techniques and secure computation methods.

Federated transfer learning is designed for scenarios where both the feature space and user samples differ across datasets but share some common elements. It applies transfer learning techniques to enable knowledge sharing between datasets with limited overlap. This is useful in cases where organizations operate in different regions or industries but seek to benefit from collective intelligence without direct data exchange.

These types of federated learning provide flexibility in training machine learning models while addressing privacy and collaboration challenges in distributed environments. As research in FL continues to progress, hybrid approaches that combine multiple types are being explored to optimize learning across diverse datasets.

2.2.2 Federated Learning in Real-World Applications

Federated Learning has gained widespread adoption across various domains, offering privacy-preserving, decentralized model training solutions. Its ability to enable collaborative learning without sharing raw data makes it particularly valuable in sensitive and distributed environments.

One of the most impactful applications of FL is in healthcare, where hospitals and research institutions can collaboratively train machine learning models without violating patient privacy regulations [35]. It is used for disease diagnosis, medical imaging, and predictive analytics, allowing institutions to benefit from a larger dataset while maintaining compliance with privacy laws such as HIPAA [36].

In the financial sector, FL enhances fraud detection, risk assessment, and anti-money laundering systems by enabling banks and financial institutions to collaborate without exposing confidential customer data [37]. By sharing model insights instead of raw transactional data, FL helps improve security and decision-making across distributed banking networks [38].

Federated learning is also transforming internet of things and edge computing, where resource-constrained devices such as smart home assistants, mobile phones, and autonomous vehicles can train models locally and contribute to global learning [3]. This reduces latency and enhances personalization while minimizing data transfer costs.

Another key application is in natural language processing and smart assistants, where FL enables privacy-focused language model training. Virtual assistants such as Google Assistant and Apple's Siri use FL to refine their models based on user interactions without sending raw voice or text data to central servers [35].

Other emerging applications include cybersecurity, where FL is used for distributed intrusion detection systems, and smart cities, where decentralized learning helps optimize traffic management and energy consumption [36]. As FL continues to evolve, its adoption across diverse domains is expected to expand, addressing key privacy and scalability concerns in data-driven applications.

2.2.3 Challenges in Federated Learning

Federated Learning presents several challenges that impact its efficiency, security, and scalability. One key issue is data heterogeneity, where client datasets are non-independent and identically distributed. This variation leads to inconsistent model updates and slows convergence, affecting overall model performance [35]. Adaptive aggregation techniques and clustering methods have been explored to mitigate these effects [36].

Another major challenge is communication overhead. Frequent model updates between clients and the server result in high bandwidth consumption, making FL inefficient in large-scale deployments, especially for mobile and internet of things devices [6]. Techniques such as model compression and asynchronous updates help reduce communication costs [39].

Despite its decentralized nature, FL remains vulnerable to security and privacy threats. Inference attacks attempt to reconstruct private data from model updates, while poisoning attacks introduce adversarial modifications that degrade performance [35]. Privacy-preserving techniques such as differential privacy and secure multi-party computation help mitigate these risks but often introduce computational overhead [36].

Scalability is another concern, as FL involves numerous clients with limited computational resources. Increasing the number of participants complicates model aggregation and can lead to slow training [6]. Lightweight models and efficient aggregation strategies are being explored to improve scalability [39].

Finally, fairness remains a challenge, as clients contribute unequally to model training, potentially reinforcing biases. Ensuring equitable learning and personalization remains an active research area [35]. Addressing these challenges is crucial for FL's broader adoption across real-world applications.

2.3 Hierarchical and Clustered Federated Learning

As FL continues to be adopted across diverse domains such as healthcare, autonomous vehicles, and edge computing, it faces persistent challenges related to communication overhead, system heterogeneity, and non-IID data distributions. Traditional flat client-server architectures often struggle to address these issues at scale.

To overcome these limitations, two complementary paradigms have emerged: Hierarchical Federated Learning (HFL) and Clustered Federated Learning (CFL). HFL introduces a multi-tiered architecture that reduces communication overhead and improves scalability by leveraging intermediate aggregators such as edge servers. In contrast, CFL groups clients with similar data distributions to train specialized models, thereby enhancing performance in heterogeneous data settings.

These paradigms reflect different directions in the evolution of HFL focuses on system-level efficiency, while CFL emphasizes statistical heterogeneity and personalization. In practice, they can also be combined to simultaneously improve scalability and model robustness [40, 41].

2.3.1 Hierarchical Federated Learning

HFL introduces an intermediate aggregation layer between clients and the central server, forming a multi-level architecture that significantly reduces communication costs and improves model convergence in distributed and resource-constrained environments. This structure is particularly suitable for scenarios such as mobile-edge-cloud systems and vehicular networks [13].

For example, Abad et al. [42] propose an HFL architecture tailored for heterogeneous cellular networks, reducing latency and enhancing scalability. Similarly, Liu et al. [43] design a client-edge-cloud hierarchy where edge servers aggregate local updates before sending them to the central server, improving robustness to network variability and client diversity.

Further optimization of training cost and energy efficiency has been studied in various works. Cui et al. [44] introduce a cost-aware optimization framework that reduces training delay in mobile-edge-cloud environments. Chen et al. [45] present HED-FL, an energy-efficient and adaptive HFL framework that dynamically adjusts based on system resource constraints.

2.3.2 Clustered Federated Learning

CFL addresses data heterogeneity by clustering clients based on data or model similarity and training separate models for each cluster. This approach is particularly effective in non-IID environments, where averaging updates from diverse clients can hinder convergence [40, 46].

Briggs et al. [46] propose a method that performs hierarchical clustering on local updates to improve model performance under non-IID data. Yan et al. [47] explore dynamic clustering strategies for heterogeneous environments, showing improvements in both convergence and generalization.

In vehicular networks, Taïk et al. [48] propose a CFL approach that forms dynamic clusters based on mobility patterns, optimizing communication and training efficiency for vehicle-to-infrastructure interactions.

In summary, while HFL targets communication scalability and system structure, CFL improves learning performance by grouping clients based on data characteristics. These two strategies are complementary and, when combined, can support more effective and scalable federated learning systems.

2.4 Federated Reinforcement Learning

FRL is an emerging research area situated at the intersection of RL and Federated Learning (FL). It aims to enable multiple agents or clients to collaboratively learn optimal decision-making policies while keeping their locally collected experience data decentralized and private.

RL is a learning paradigm in which agents interact with an environment to learn optimal actions through trial-and-error, guided by reward signals. Traditional RL methods typically assume access to a centralized training process, where data, in the form of trajectories or experience tuples, is collected and processed in a unified manner. However, in many real-world scenarios, such as robotics, autonomous systems, or edge computing, agents operate in distributed environments where centralizing data is either infeasible or undesirable due to privacy concerns, communication overhead, or system heterogeneity.

In contrast, FL enables collaborative training across decentralized clients by exchanging model updates instead of raw data. This framework preserves data privacy while allowing knowledge sharing across distributed nodes.

FRL combines the sequential decision-making capabilities of RL with the privacy-preserving and distributed learning characteristics of FL. In this framework, each agent interacts with its local environment and learns from its own experience while periodically communicating with a central server or peer agents to aggregate knowledge. This integration allows for collaborative policy learning without sharing sensitive interaction data, paving the way for scalable and privacy-aware RL in decentralized settings [49–51].

This collaborative learning paradigm can be realized through various architectural designs, each influencing the trade-offs between scalability, efficiency, and privacy in FRL systems.

2.4.1 Federated Reinforcement Learning Architectures

FRL systems can be implemented in various architectural forms, each suited to different application domains and system constraints. Broadly, FRL architectures fall into three categories: centralized server-agent models, peer-to-peer models, and hierarchical federated architectures.

- **Centralized FRL:** In this architecture, a central aggregator coordinates learning by collecting model parameters or gradients from distributed agents. Each agent independently interacts with its local environment and periodically shares its model updates with the central server, which aggregates them and redistributes a global model. This is the most widely studied architecture due to its simplicity, though it may suffer from a single point of failure and high communication overhead [51].

- **Peer-to-Peer FRL:** In decentralized settings, agents exchange model updates directly with each other without relying on a central server. This structure improves fault tolerance and scalability but complicates aggregation due to the lack of centralized control. Consensus-based methods or gossip protocols are commonly used for coordination [50].

- **Hierarchical FRL:** This architecture introduces an intermediate layer of edge servers that aggregate local updates from nearby agents before passing them to a global server. Inspired by Hierarchical Federated Learning (HFL), this structure is particularly suited to large-scale and latency-sensitive applications such as autonomous vehicles and edge IoT networks. It reduces communication costs and improves scalability while preserving data privacy [13,52].

Each architecture introduces trade-offs between communication efficiency, fault tolerance, and convergence speed. The choice of FRL architecture depends on factors such as the number of agents, network bandwidth, system heterogeneity, and latency constraints.

2.4.2 Recent Advances in FRL

FRL has seen significant progress in recent years, addressing key challenges related to communication efficiency, agent heterogeneity, privacy preservation, and convergence guarantees.

One important development is the use of asynchronous learning frameworks. Traditional FRL methods often rely on synchronous updates, which can be delayed by slower agents. The Asynchronous Federated Policy Gradient (AFedPG) algorithm mitigates this by introducing a delay-adaptive lookahead technique, enabling more efficient updates and linear speedups proportional to the number of agents [52].

To handle agent heterogeneity, the Federated Heterogeneous Q-Learning (FedHQL) algorithm allows agents with differing architectures to collaborate without requiring a unified policy representation. This improves sample efficiency by enabling diverse agents to share knowledge while maintaining their unique models [53].

In terms of privacy-preserving personalization, the FedRLHF framework enables agents to incorporate human feedback locally while preserving data privacy. It supports personalized policy learning and offers theoretical guarantees of convergence [54].

Understanding the convergence behavior of FRL algorithms is also crucial. Recent studies have provided finite-time convergence analyses for on-policy FRL with linear function approximation, offering insights into collaboration dynamics and the effects of system heterogeneity on learning efficiency [55].

In cloud robotic systems, FRL supports knowledge fusion and transfer among distributed robots.

The Lifelong Federated Reinforcement Learning (LFRL) architecture facilitates this process, improving navigation and decision-making through shared experience without compromising data privacy [51].

These advancements collectively contribute to the maturation of FRL, pushing the boundaries of what is possible in decentralized, privacy-sensitive environments.

2.4.3 Applications of FRL

FRL has been successfully applied across a range of domains, enabling collaborative decision-making in settings where privacy and decentralization are essential. In the Internet of Things (IoT), FRL enables intelligent decision-making across distributed devices without the need for centralizing data. Pinto Neto et al. highlight use cases in security, sustainability, vehicular networks, and industrial services [49].

In autonomous vehicle systems, FRL facilitates cooperative driving behaviors. Asadi et al. propose privacy-preserving FRL techniques for managing coordination at signalized intersections, resulting in improved traffic efficiency and safety [56].

In cloud robotics, FRL enables robots to learn collaboratively from distributed interactions. The LFRL architecture enhances performance in navigation tasks by enabling robots to share and reuse knowledge while maintaining privacy [51].

In edge computing scenarios, FRL supports efficient resource allocation and task scheduling by allowing edge devices to learn collaboratively without transmitting local data, thus improving performance and reducing latency [57].

In Open Radio Access Network (O-RAN) environments, FRL is used to manage network slicing. It coordinates multiple applications to dynamically allocate communication resources, achieving higher throughput and reduced latency [58]. Zhang et al. propose a federated deep reinforcement learning framework that dynamically assigns resources to different services while preserving data privacy [14].

Beyond vehicular systems and cloud robotics, FRL has been explored in several emerging domains. In healthcare, FRL enables privacy-preserving treatment policy learning across hospitals.

Each hospital trains its own policy using patient data while benefiting from collective learning without exposing sensitive records, thereby ensuring compliance with regulations like HIPAA [35].

In finance, FRL has been utilized for distributed fraud detection and dynamic portfolio optimization. By training policies collaboratively across banks or trading platforms, these systems improve detection capabilities while keeping customer transaction histories confidential [17].

Smart grid and energy systems also benefit from FRL. In this context, agents such as smart meters or distributed energy resources learn control policies to balance power load, optimize energy consumption, and support demand response in real time. FRL facilitates policy learning across geographically distributed systems while preserving data locality [58].

These applications highlight the growing versatility of FRL in privacy-sensitive, real-time, and distributed environments where centralized RL is impractical or prohibited.

2.4.4 Challenges and Open Issues

While FRL offers significant promise, it also introduces a range of challenges and unresolved issues. A primary concern is the heterogeneity of local environments and data distributions, which leads to non-IID scenarios [35, 36]. This complicates the aggregation of local models and can degrade global policy performance. Additionally, frequent communication between agents and the server, necessary for iterative RL, can result in significant bandwidth consumption and latency, especially in large-scale systems.

Privacy and security also remain pressing concerns. Although FL avoids transmitting raw data, model updates themselves can leak sensitive information through inference attacks. Ensuring robust differential privacy and secure aggregation techniques in the context of continuous, interaction-driven learning remains a critical research area.

Fault tolerance is another practical concern in FRL, where some agents may drop out or send unreliable updates due to unstable network conditions or hardware limitations. Fan et al. propose a fault-tolerant FRL framework that maintains theoretical convergence guarantees even when a fraction of agents fail or behave unpredictably, making FRL more robust in real-world deployments [59].

Another open issue is the design of effective incentive mechanisms. Since participation in FRL may incur costs related to computation, energy, or bandwidth, agents may act selfishly or drop out if not properly incentivized. Recent surveys have proposed various economic and game-theoretic strategies, including Stackelberg games, auctions, and contract theory, to address this issue in FL [60, 61]. However, adapting these mechanisms to the dynamic and long-term learning process of FRL remains a challenge.

A recent line of work has begun addressing these gaps by incorporating economic theory into FRL under multi-round and dynamic participation settings. For instance, contract-theoretic approaches have been proposed to design reward mechanisms that ensure incentive compatibility (IC) and individual rationality (IR), while accounting for agents' strategic behavior and non-uniform data growth across training rounds. These models formulate convex optimization problems to guarantee rational engagement, even as agent contributions evolve over time. However, existing works primarily focus on single-layer FRL setups and overlook budget constraints or cross-layer coordination, leaving the challenge of hierarchical incentive design in FRL largely unaddressed [12].

Addressing these challenges requires interdisciplinary collaboration across machine learning, systems engineering, economics, and privacy-preserving technologies to develop FRL frameworks that are scalable, reliable, and equitable. In this context, our work leverages a clustered FRL framework to directly address the dual challenges of environment diversity and limited server budget through principled incentive design. By integrating contract theory and Stackelberg games, we propose a scalable and incentive-compatible solution tailored for real-world deployment.

2.5 Incentive Mechanisms

Incentive mechanisms have become increasingly essential in distributed learning systems such as Federated Learning (FL), where autonomous and potentially self-interested agents are expected to contribute computational resources, data, or both. Without adequate motivation, participants may act selfishly, provide low-quality data, or even withhold participation altogether. This issue is not unique to FL; similar concerns have been extensively studied in participatory sensing, mobile crowd sensing, and Internet of Things (IoT) environments. Restuccia et al. emphasize that incentive design is fundamental to sustaining engagement in participatory sensing systems, especially when there are costs or risks involved for contributors [62]. Likewise, Jaimes et al. outline how incentive techniques are critical in mobile crowd sensing, where users' willingness to participate fluctuates based on perceived rewards or benefits [63]. Maddikunta et al. provide a broad overview of incentive strategies in the IoT domain, highlighting how economic, social, and reputation-based incentives can influence cooperative behavior among distributed devices and human users [64]. From a theoretical perspective, Ratliff et al. argue that incentive design lies at the heart of many distributed optimization and control problems, and that a careful balance must be struck between system efficiency, fairness, and strategic behavior [65]. Collectively, these insights underscore the importance of designing robust and context-aware incentive mechanisms to ensure sustainable and effective collaboration in federated and decentralized systems.

2.5.1 Incentive Mechanisms in FL

As FL relies on the voluntary participation of distributed agents such as mobile devices, edge nodes, or organizations, designing effective incentive mechanisms becomes critical to ensure sustained and meaningful collaboration. In contrast to centralized machine learning, FL introduces new complexities due to its decentralized nature, the heterogeneity of participants, and the cost associated with data sharing, computation, and communication. Without appropriate incentives, participants may contribute subpar models, manipulate learning outcomes, or refuse to participate altogether.

Recent literature has explored various economic and game-theoretic approaches to encourage agent participation in FL. Tu et al. provide a broad taxonomy of these approaches, categorizing them into auctions, Stackelberg games, contract theory, and reinforcement learning-based designs, each tailored to specific assumptions about agent behavior and system requirements [66]. Zhan et al. provide a comprehensive survey that highlights both the design objectives, including fairness, efficiency, truthfulness, and privacy, and the unique challenges inherent in FL environments [67]. In a separate work, Zhan et al. delve into practical issues and propose high-level design principles that balance participant motivation and system-level goals [17].

Beyond surveys, recent work has also examined the theoretical foundations of incentive design in FL. Huang and Wang present a unified framework grounded in economic and game-theoretic models to analyze strategic interactions among agents, focusing on both cooperative and non-cooperative settings [68]. Ahmed et al. conduct a systematic review of existing incentive mechanisms and their associated security challenges, highlighting the need for trustworthy mechanisms that are robust to adversarial behaviors and incentive manipulation [69]. Moreover, Zhao et al. explore the use of multi-agent RL for designing adaptive incentive schemes in multi-task federated edge learning scenarios, underscoring the growing trend of leveraging learning-based approaches in this space [70].

Together, these studies paint a diverse and evolving landscape for incentive mechanisms in FL, reflecting the interdisciplinary nature of the field. They provide a strong foundation for deeper exploration into specialized frameworks such as contract theory and Stackelberg games, which are examined in the following subsections.

2.5.2 Contract Theory-Based Incentives

Contract theory is a fundamental framework from economics that studies how agreements can be designed between parties with potentially conflicting interests, particularly under conditions of asymmetric information. In the context of incentive mechanism design, contract theory provides structured models to align the incentives of self-interested agents (e.g., clients in FL) with those of a central authority (e.g., model owner or aggregator). As outlined in the foundational work by Bolton and Dewatripont, contract theory revolves around designing agreements that ensure participants voluntarily act in a way that leads to desirable outcomes, even when their private information is hidden from the principal [71].

A contract typically consists of a menu of offers, each tailored to different types of agents based on characteristics such as data quality, computational capacity, or communication cost. In Federated Learning, these agent types are often not known a priori by the aggregator, which introduces information asymmetry into the system. To handle this, the principal (aggregator) must design incentive-compatible contracts, where each agent voluntarily selects the contract intended for its type. This mechanism design ensures that high-quality or high-contribution clients receive appropriate rewards while discouraging misreporting or strategic manipulation. Zhang et al. provide a broad overview of contract-theory-based mechanisms in wireless networks, which serve as a natural foundation for FL scenarios where resource heterogeneity and participation costs are prevalent [72].

Several studies have applied contract theory to FL in various domains. Kang et al. propose an incentive mechanism for mobile FL systems where the aggregator cannot directly observe the local training cost or data quality of participants. Their contract-based solution optimally maps expected contributions to monetary rewards under budget constraints [73]. Similarly, Saputra et al. adopt contract theory to incentivize electric vehicle networks in FL settings. Their economic-efficiency framework captures both agent heterogeneity and the time-varying nature of contribution, showing improved utility for both the aggregator and participants [74]. Kang et al. further enhance reliability by combining reputation systems with contract design, enabling more robust participation selection in the presence of untrusted or inconsistent clients [75].

Li et al. develop a contract-based scheme tailored to health crowdsensing, where clients' data quality and energy cost are private. Their model ensures that users are incentivized to reveal their types truthfully and that the aggregator achieves utility maximization under practical constraints [76]. Similarly, Lim et al. propose a dynamic contract model for smart healthcare applications, where the aggregator updates contracts over time based on observed performance and evolving requirements [16]. These dynamic models are particularly suited to FL systems where client performance may vary across rounds due to mobility, hardware changes, or network fluctuations.

Recent works have also extended contract theory to more complex FL environments. Wang et al. design a contract mechanism for clustered FL, where clients are grouped into clusters and offered group-specific contracts to balance global efficiency and local heterogeneity [77]. Yang et al. apply contract theory to asynchronous FL settings, designing incentives that accommodate non-synchronous client participation while maintaining truthfulness and fairness [78]. These efforts reflect the growing trend of adapting contract theory to meet the diverse operational and technical constraints in real-world FL deployments.

Finally, recent reviews underscore the importance of contract theory in designing secure and trustworthy incentive systems in FL. Ali et al. and Lim et al. both emphasize that contract-theoretic approaches provide robust models to handle information asymmetry, strategic misreporting, and long-term engagement, especially in mobile and resource-constrained environments [79, 80]. By aligning agent incentives with system-level goals through well-designed contracts, these mechanisms offer a practical and theoretically sound path toward sustainable participation in Federated Learning.

2.5.3 Stackelberg Game-Based Incentives

Stackelberg game theory is a hierarchical decision-making framework in which a leader makes a strategic move first, and followers respond after observing the leader's decision. This sequential structure captures many real-world scenarios involving strategic interactions where one party (typically with more authority or information) influences the behavior of others through incentive design. Unlike simultaneous-move games, Stackelberg games model the temporal and hierarchical structure of decisions, allowing the leader to anticipate the optimal response of followers. This framework has been widely adopted in various domains, including smart grids [81], wireless networks [82], and economic systems [83], due to its ability to represent asymmetric power relationships and sequential dependencies.

The basic components of a Stackelberg game include a set of players classified into leaders and followers, strategy spaces for each player, and payoff functions that depend on the chosen strategies. The leader selects a strategy to maximize its own utility, anticipating the best response (i.e., the follower's Nash equilibrium strategy) for every possible action it might take. The solution concept is known as the Stackelberg equilibrium, which is a refinement of the subgame perfect Nash equilibrium in dynamic games [84]. In the context of Federated Learning (FL), this hierarchy maps naturally to the principal–agent relationship: the FL server acts as the leader offering incentive schemes, while the clients act as rational followers who choose their participation levels based on the proposed incentives.

Sarikaya and Ercetin were among the first to apply Stackelberg game models to FL. They model the FL server as a leader that sets payments to mobile users (followers) to motivate their participation in training while minimizing its own cost. The users respond by selecting effort levels that balance energy consumption and reward, resulting in a Stackelberg equilibrium that ensures mutually beneficial outcomes [85]. More recent work expands on this foundation to address practical complexities in FL environments. For instance, Xiao et al. design a two-stage Stackelberg game where the server first selects an incentive budget and then allocates individual rewards based on participants' declared costs and expected contributions. Their model ensures incentive compatibility and fairness while optimizing global utility [86].

In UAV-enabled edge computing scenarios, Stackelberg games have been employed to manage clustered federated learning systems. He et al. develop a three-stage Stackelberg game that involves a server (leader), cluster heads, and UAV agents (followers) with varying energy levels and communication capacities. Their model demonstrates the effectiveness of hierarchical learning and decision-making under strict resource constraints [87]. Similarly, Zhou et al. apply Stackelberg game models to FL in mobile edge networks, showing how a server can allocate communication and computation resources to heterogeneous UAVs while maintaining system efficiency [88].

In the context of IoT systems, where devices exhibit diverse capabilities and operate under limited energy budgets, Stackelberg games provide a structured approach to incentivize participation. Chen et al. propose a multifactor Stackelberg game model that incorporates user preferences, task complexity, and device reliability to design personalized incentives that align local decision-making with global learning objectives [89]. Zeng et al. also present a broader game-theoretic framework for incentive design in FL, identifying Stackelberg models as particularly suited for leader–follower dynamics where the server possesses more information and control over the learning pipeline [90].

Overall, Stackelberg game theory provides a powerful and flexible tool for modeling incentive interactions in FL, especially when agents are rational, resource-constrained, and strategic. Its hierarchical structure aligns well with the operational reality of FL, where a centralized server interacts with multiple decentralized clients. By leveraging the anticipation of follower responses, Stackelberg-based mechanisms enable optimal incentive allocation, improved fairness, and greater system efficiency. The equilibrium-driven approach also facilitates formal analysis and the design of robust, scalable incentive schemes suitable for dynamic, heterogeneous learning environments.

2.5.4 Comparison and Discussion

Contract theory and Stackelberg games represent two of the most widely adopted frameworks for incentive mechanism design in Federated Learning (FL), each with unique strengths. Contract theory is particularly effective in scenarios with asymmetric information, where the server cannot observe agents' private characteristics, such as data quality or computational cost. By offering a menu of contracts, the server induces agents to self-select into contracts that match their types. This promotes truthful behavior and ensures incentive compatibility. For example, Kang et al. and Saputra et al. apply contract theory to design efficient and fair incentives in mobile and energy-constrained FL environments [73, 74]. However, contract design may become complex as the number of agent types increases or when user preferences change dynamically over time.

Stackelberg game models, on the other hand, emphasize sequential interactions between a leader (the server) and followers (the clients). These models are particularly suitable when the server can anticipate client responses and optimize its strategy accordingly. Sarikaya and Ercetin formulate one of the earliest Stackelberg-based FL incentive models, where the server sets reward levels and clients respond with participation decisions based on energy costs [85]. More recent studies explore multi-stage Stackelberg games and their integration into clustered and mobile edge FL systems [87, 88]. While Stackelberg models are powerful in capturing temporal dynamics and strategic anticipation, they often assume rational behavior and full observability of client reactions, which may not hold in real-world deployments.

Beyond these two frameworks, other incentive mechanisms have been explored in the FL literature. Auction-based approaches treat the selection of participants as a bidding process, where clients submit their participation costs and the server selects those that offer the best utility under budget constraints. These models, such as Vickrey–Clarke–Groves (VCG) auctions or reverse auctions, promote truthfulness and resource efficiency but require strong assumptions on bidding behavior and communication overhead. Shapley value-based approaches, rooted in cooperative game theory, offer a fairness-oriented mechanism by allocating rewards proportionally to each client's marginal contribution to the global model. Zhan et al. and Shamsian et al. propose methods that approximate the Shapley value in FL to ensure fairness, though the computational cost of exact evaluation remains a significant barrier [15, 67].

Each of these mechanisms has trade-offs. Stackelberg and contract-theoretic methods offer strong theoretical guarantees and flexibility, but require careful modeling of agent preferences and system dynamics. Auction-based approaches are simple and scalable but may not capture long-term participation incentives. Shapley value methods promote fairness but suffer from scalability issues.

Ultimately, the choice of incentive mechanism depends on the FL system's goals, constraints, and assumptions about agent behavior. In this thesis, we focus on contract theory and Stackelberg games due to their theoretical richness and suitability for modeling personalized participation in FRL, subject to constraints on resources, budget, and compensable incentives.

Despite the growing body of work on incentive mechanisms in Federated Learning, challenges remain in designing mechanisms that are robust, theoretically grounded, and effective under strict budget constraints on the main server, particularly in dynamic and heterogeneous environments. These challenges are further amplified in FRL, where agents operate in heterogeneous environments, interact with evolving policies, and incur varying computational costs. This thesis addresses these gaps by exploring incentive-compatible mechanisms tailored for FRL, leveraging tools from contract theory and Stackelberg game models. Building on these foundations, our proposed framework unifies contract-theoretic local incentives and Stackelberg-based global budget allocation into a coherent two-layer mechanism. This design explicitly considers strategic agent behavior, information asymmetry, and hierarchical system dynamics, gaps that remain underexplored in current literature.

Chapter 3

System Model and Problem Formulation

This chapter presents the methodological foundation of the proposed framework for incentivizing participation in FRL. As discussed in Chapter 2, two key challenges in real-world FRL systems are the heterogeneity of local environments and the scalability of coordination among numerous agents. To address these issues, we adopt a clustered FRL architecture commonly used in the literature, where agents are grouped based on environmental similarity and organized under a hierarchical structure to facilitate scalable learning. Another critical challenge is the main server's budget for rewarding participants, which necessitates a cost-effective incentive mechanism that ensures agent cooperation while optimizing overall system performance. In this chapter, we first describe the system model, including the hierarchical and clustered FRL setting. Then, we formally formulate the incentive design problem using a contract-Stackelberg approach, which leverages contract theory and Stackelberg game theory in a hierarchical manner to model the strategic interactions between the main server and local servers, as well as between local servers and agents, under budget constraints and asymmetric information.

3.1 System Model

Consider a clustered FRL framework composed of one main server, multiple local servers, and a set of distributed agents, as illustrated in Fig. 3.1. The system operates in two hierarchical layers: the lower layer, where agents interact with local servers, and the upper layer, where local servers

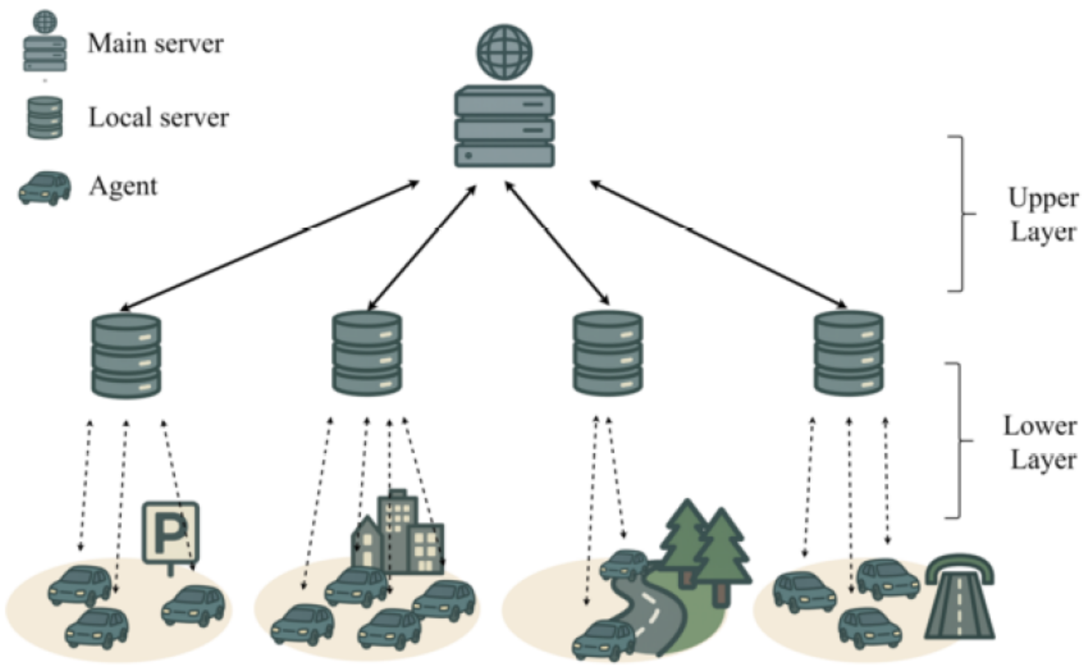


Figure 3.1: Clustered federated reinforcement learning framework with a two-layer hierarchical structure.

communicate with the main server.

We define each communication period between a local server and its agents (lower layer) as a round, and each communication period between the main server and a local server (upper layer) as an iteration.

At the agent level, each agent trains an RL model to determine the optimal action for every possible state. The local RL model follows the standard Bellman equation and is updated by solving the following equations

$$Q(s,a) = R(s,a) + \gamma_{disc} \sum_{s'} P(s' \mid s,a) \max_{a'} Q(s',a'),$$

$$a^* = \operatorname*{argmax}_{a} Q(s,a).$$

Here, $Q(s,a)$ represents the action-value function, $R(s,a)$ is the immediate reward, γ_{disc} is the discount factor, and $P(s' \mid s,a)$ denotes the transition probability between states.

We consider a clustered structure at the lower layer, where each cluster contains a local server and multiple agents. The local server initially distributes a local model to its agents, who then train the model by interacting with their environment and collecting data. After training, the agents send their updated models back to the local server for aggregation. This process repeats iteratively within the cluster until the model reaches a desired level of accuracy.

For agent k in round l, its dataset, denoted as $D_k^l = \{(\chi_{k,n}^l, y_{k,n}^l)\}_{n=1}^{S_k^l}$, consists of data samples $\chi_{k,n}^l$ and their corresponding labels $y_{k,n}^l$, where S_k^l represents the number of data samples available to agent k in round l. The loss function for training the local model of agent k is given by

$$L_k^l(\psi_k^l) = \frac{1}{S_k^l} \sum_{n \in S_k^l} \left| y_{k,n}^l - \hat{y}_{k,n}^l \right|,$$

where ψ_k^l denotes the model parameters for agent k in round l, and $\hat{y}_{k,n}^l$ is the predicted output.

Once local training is completed, agents transmit their updated model parameters to the local server for aggregation. The aggregated cluster model for cluster m in round l is obtained by optimizing

$$\psi_m^l = \operatorname{argmin} \sum_{k=1}^{N_m} \frac{S_k^l}{S_m^l} L_k^l(\psi_k^l),$$

where N_m denotes the number of agents in cluster m, and S_m^l represents the total number of data samples within cluster m during round l.

At the upper layer, after training is completed within each cluster, the local servers transmit their aggregated models to the main server for further consolidation. The global model parameters are then updated by aggregating the cluster-level models according to

$$\psi^* = \operatorname{argmin} \sum_{m=1}^{M} \frac{S_m^l}{S^{\text{total}}} L_m^l(\psi_m^l),$$

where M is the total number of clusters, S_m^l is the data size of cluster m, and S^{total} is the total number of data samples across all clusters.

The main server then updates the global model and redistributes it back to the local servers as the initial model for the next training phase. This two-tier iterative process continues until the final

Table 3.1: Important Notations

Notation	**Description**
$\psi_k^l, \psi_m^l, \psi^*$	Local model parameters for agent k in round l, cluster level model parameter for cluster m in round l and the global model parameter.
θ_k	Computation cost of agent k and its type in contract theory.
Π, π_m	Utility functions of the main server and local server m in the upper layer.
$U_m, u_{m,i}^l$	Utility functions of the local server m and type-i agent in round l, within cluster m at the lower layer.
T, B_m	Total Budget from the main server to local servers, and budget for local server m.
A_m	Accuracy gain by local server m.
γ_m	Fraction of B_m allocated to agents in cluster m.
M	Number of clusters.
$\alpha_m^l, \delta_{m,i}^l$	Portion of $\gamma_m B_m$ allocated to round l, and the portion of the budget allocated to type-i agents in round l within cluster m.
$N_m, p_{m,i}$	The total number of agents in the cluster m, and the probability distribution of agent types.
$q_{m,max}^l, q_i^l$	Maximum permissible data contribution in round l, and data by type-i agent in round l within cluster m.
R_i^l, R_i^{l*}	Reward for type-i agent in round l and its optimal value.

model, generated by the main server, achieves the desired accuracy.

Performing a reinforcement learning task imposes computational costs on agents [12]. For agent k in round l, the computation energy cost is defined as

$$e_k^l(f_k) = \eta_k \nu_k S_k^l f_k^2,$$

where η_k is the effective capacitance parameter of the computing chipset for agent k, ν_k is the number of CPU cycles required by agent k to process one data sample, and f_k represents the computational resource allocated by agent k. Additionally, the computation time can be calculated as

$$\Delta_k^l = \frac{\nu_k S_k^l}{f_k}.$$

Each agent has a local data accuracy or reliability requirement, represented by ϵ_k. The number of local iterations needed to achieve this accuracy is given by $\log\left(\frac{1}{\epsilon_k}\right)$. Consequently, the total energy cost for the agent k in each training round can be expressed as

$$E_k^l = \log\left(\frac{1}{\epsilon_k}\right) e_k^l(f_k).$$

Furthermore, the unit computation cost for agent k, denoted by θ_k, is defined as

$$\theta_k = \frac{\log\left(\frac{1}{\epsilon_k}\right) e_k^l(f_k)}{S_k^t}.$$

The computation energy cost model $e_k^l(f_k) = \eta_k \nu_k S_k^l f_k^2$ is rooted in CMOS circuit theory, where dynamic power is proportional to the square of the clock frequency [91]. It realistically captures energy costs for processing S_k^l samples with ν_k CPU cycles each, under frequency f_k and capacitance η_k. This modeling is standard in energy-aware RL and digital system design, justifying its use for agent-level cost characterization.

Given these non-negligible computation costs, a fundamental challenge is to incentivize agents to participate in training. To address this, the main server allocates a limited budget to local servers at the beginning of each iteration. Each local server must then determine how to distribute this budget among its agents and across future rounds.

Given that the main server must distribute a budget across both layers of the hierarchy, the incentives available to agents are inherently constrained. Thus, an effective incentive mechanism is required to maximize participation while ensuring efficient budget utilization. At the lower layer, agents should be motivated to contribute through appropriate rewards, while at the upper layer, local servers must strategically allocate their share of the budget to incentivize agent cooperation and improve model accuracy.

The key notations used throughout this paper are summarized in Table 3.1.

3.2 Problem Formulation

In this section, we mathematically formulate the problem for each layer of the system. Additionally, a budget allocation mechanism must be designed to effectively balance model accuracy and agent incentives. This allocation is structured based on the problem's inherent constraints and characteristics. We first analyze the budget constraints imposed by the main server before developing the reward mechanisms.

At the lower layer, the local server must reward its agents based on their contributions while considering its budget constraints. The reward system is crucial, as it directly impacts both the local server's performance (in terms of model accuracy) and the agents' motivation to participate.

At the upper layer, the main server acts as the leader, determining a total budget T to be distributed among the local servers at the start of each iteration. The local servers, as followers, then decide how to allocate this budget, balancing between rewarding agents for their contributions to model training and retaining a portion as profit. While local servers may prefer to keep a larger share of the budget, doing so could compromise training accuracy within their clusters. Poor training accuracy may, in turn, lead to reduced budget allocation from the main server in subsequent iterations.

The challenge is to determine the optimal strategy for local servers that maximizes their utility while maintaining sufficient model accuracy. Notably, the leader's budget allocation and the followers' strategies evolve in each iteration based on available resources, making the optimization dynamic.

3.2.1 Budget Constraint

At the start of each iteration, the main server allocates a budget B_m to the local server of cluster m, which then determines how to distribute a portion, $\gamma_m B_m$, to support training and incentivize agent participation. Here, γ_m represents the fraction of B_m dedicated to agent incentives, while the remaining portion is retained as profit by the local server.

This budget is further allocated across rounds and agents. We define $B^l_{m,i}$ as the budget assigned to type-i agent in round l. To reflect the incremental nature of data contribution, the local server initially assigns a smaller portion to earlier rounds and increases the allocation in later rounds. This budget distribution plays a crucial role in agent engagement and overall model performance within the cluster.

To achieve this, we introduce a new variable α_m^l, which represents the proportion of the budget that the local server m allocates to round l. Given the nature of FRL, data contributions increase after each round, so α_m^l should be an increasing function along the rounds. We define α_m^l as

$$\alpha_m^l = \frac{q_{m,\text{max}}^l}{\sum_{k=l}^{L} q_{m,\text{max}}^k}, \tag{1}$$

where $q_{m,\text{max}}^l$ denotes the maximum amount of data that each agent can contribute to the local server m at round l. This value varies across different clusters, rounds, and agent types. In practice, each agent continuously explores the environment and collects experience in the form of state-action pairs. However, since each round is constrained by a limited time interval, the number of interactions processed by each agent for local training in each round is capped by $q_{m,\text{max}}^l$.

As time progresses and agents accumulate an increasing number of state-action pairs through interaction with the environment, the value of $q_{m,\text{max}}^l$ gradually increases, enabling local models to be trained using a larger set of experiences. In this paper, we assume that all agents interact with the environment and collect experience at the same rate. We also consider a distinct q_{max} for each agent, meaning that $q_{m,\text{max}}^l$ is represented as a vector of I_m elements, where each element corresponds to the maximum allowed data contribution for a specific agent type within the cluster m in round l. The values in this vector may vary across different rounds.

From (1), α_m^l is dynamically adjusted based on the remaining rounds. Thus, this definition can help balance the budget allocation, especially in cases where contributions vary significantly across rounds.

At the start of each round, the available budget is calculated as follows

$$B_m^l = \begin{cases} \alpha_m^l \cdot \gamma_m \cdot B_m, & \text{if } l = 1, \\ \alpha_m^l \cdot \left(\gamma_m \cdot B_m - \sum\limits_{k=1}^{l-1} \sum\limits_{j=1}^{I_m} p_{m,j} N_m R_j^k\right), & \text{if } l \neq 1. \end{cases} \tag{2}$$

Equation (2) ensures that the total budget across all rounds sums to $\gamma_m B_m$.

To determine budget distribution among the available agent types, we introduce $\delta_{m,i}^l$ to represent the share of the budget that each type-i agent will receive in round l within cluster m. Since budget allocation depends on data contribution, we define $\delta_{m,i}^l$ as follows

$$\delta_{m,i}^l = \frac{p_{m,i} N_m q_i^l}{\sum_{j=1}^{I_m} p_{m,j} N_m q_j^l}, \tag{3}$$

where q_i^l represents the data contribution of type-i agent at round l, and $p_{m,i} N_m$ is the number of type-i agents. Obviously, equation (3) implies that agent types with higher data contributions receive a larger share of the budget.

Thus, given the total available budget B_m^l for each round l, the received budget for each agent type equals

$$B_{m,i}^l = \delta_{m,i}^l \cdot B_m^l. \tag{4}$$

3.2.2 Contract Design

In each cluster, a local server coordinates multiple agents who participate in model training. Agents in cluster m belong to I_m different types, each characterized by a private computation cost θ_i, which is unknown to the local server. The asymmetry of information between the local server and agents makes it challenging to directly assign rewards based on true agent costs. To address this issue, we employ contract theory, which provides a principled approach for designing incentive mechanisms under asymmetric information. By offering a menu of contract items tailored to different agent types, the local server can induce self-selection and align agent behavior with the cluster's learning objectives.

Without loss of generality, we assume an ascending order of agent types, i.e., $1 \leq \cdots \leq \theta_i \leq \cdots \leq \theta_{I_m}$ where type I_m agents incur the highest computation cost. The contract formulated by the model owner in a multi-period setting is denoted as $C(\theta_i) = \{c_i^l : 1 \leq l \leq L, 1 \leq i \leq I_m\}$, where c_i^l represents the contract designed for a type-i agent at round l. Each contract c_i^l consists of the required amount of data from each agent type and a reward as the incentive for cost compensation, which can be defined as $c_i^l = \left(q_i^l, R_i^l\right)$. Based on these definitions, we can derive the utility of type-i

agent in round l as follows

$$u_i^l(c_i^l, \theta_i) = R_i^l - \theta_i q_i^l. \tag{5}$$

After each round, the local server gathers contributions from all participating agents and aggregates them to update the global model in the federated learning setting. The utility of the local server m in round l can be defined as follows

$$U_m^l = \sum_{i=1}^{I_m} p_{m,i} N_m (\sigma \log(1 + \eta q_i^l(B_{m,i}^l)) - R_i^l). \tag{6}$$

Following [16], we used a concave function to model the diminishing effect of data quantity on the accuracy of the model in local server. Here, $p_{m,i}$ represents the probability that an agent belongs to type i among the N_m available agents. The parameters σ and η are calibration constants that reflect the impact of accuracy on utility and data quantity on accuracy, respectively. The utility of the local server m over an entire iteration, consisting of L rounds, can be expressed as

$$U_m = \sum_{l=1}^{L} U_m^l. \tag{7}$$

3.2.3 Stackelberg Game Framework

We define the utilities for the main server ($\Pi(T)$) and the local servers ($\pi_m(\gamma_m, \gamma_{-m})$) in the upper layer, respectively. By considering the sequential decision-making between the main server and local servers, we model the incentive mechanism as a two-stage Stackelberg game. The main server, acting as the leader and determining the total budget T, seeks to maximize its utility by optimally allocating T to incentivize local servers to contribute higher accuracy. In response, each local server, as a follower, seeks to maximize its own profit by strategically selecting a parameter γ_m, which governs how it distributes its allocated share among its agents, balancing between conserving resources and delivering sufficient accuracy to the main server.

Since the local servers' utilities are dependent on the main server's decision, and both the main server and the local servers aim to maximize their respective objectives during each training iteration, the interaction naturally aligns with the Stackelberg game framework.

The utility of the main server is defined as the gain in model accuracy minus the total budget paid to the local servers. As shown in [92], the accuracy of the training model can be regarded as a concave function of the accuracy gain provided by the local servers, which itself is a concave function of the data contributed during the training process in RL. Therefore, the utility of the main server for one iteration, can be expressed as

$$\Pi(T) = \xi \ln \left(1 + \sum_{m=1}^{M} A_m(\gamma_m B_m) \right) - T, \tag{8}$$

where ξ is a system parameter, and the first term represents the accuracy gain achieved collectively by all local servers. A_m denotes the accuracy gain provided by the data contributed by agents in cluster m during their training process. This accuracy depends on the parameter $\gamma_m B_m$, where B_m is the total received budget from the main server, and γ_m is the portion of budget investing for model training within the cluster m.

In a single iteration, the local servers act as followers. Considering that the reward received by a local server is proportional to the accuracy it contributes, the share of T allocated to local server m is given by

$$B_m = \frac{A_m}{\sum_{i=1}^{M} A_i} \cdot T, \tag{9}$$

where, the accuracy gain A_m provided by a local server m is given by

$$A_m = \sum_{l=1}^{L} \sum_{i=1}^{I_m} p_{m,i} N_m \big(\sigma_m \log(1 + \eta_m q_i^l(B_{m,i}^l)) \big), \tag{10}$$

where $B_{m,i}^l$ denotes the previously defined budget allocated to type-i agents by local server m in round l, which governs their data contribution as discussed earlier. The utility of local server m in the current iteration is then expressed as

$$\pi_m(T, \gamma_m, \gamma_{-m}) = (1 - \gamma_m) \cdot B_m, \tag{11}$$

where $\gamma_{-m} = (\gamma_1, \gamma_2, \ldots, \gamma_{m-1}, \gamma_{m+1}, \ldots, \gamma_M)$ denotes the strategies of all other local servers except m.

Obviously, the interaction between the two layers is governed by budget constraints, whereby the allocation decisions made at the upper layer (i.e., the main server distributing its total budget among local servers) directly shape the incentive strategies available to local servers. These local strategies, in turn, determine how agents at the lower layer are rewarded, ultimately influencing their level of participation and contribution to model training.

Chapter 4

Contract-Stackelberg Game Framework

To address the key challenges in FRL, namely, environmental heterogeneity, information asymmetry, the design of hierarchical incentive mechanisms, and budget limitations, we propose a two-layer incentive framework that integrates contract theory and Stackelberg game theory. This framework operates across two layers: at the lower layer, local servers employ contract-theoretic models to personalize rewards under asymmetric information and budget limitation; at the upper layer, the main server utilizes a Stackelberg game to strategically allocate its budget across clusters. This design ensures incentive compatibility, efficient budget distribution, and sustained agent engagement. The remainder of this chapter details the contract formulation for local servers, the budget optimization strategy at the global level, and the coordinated algorithm that integrates both layers.

4.1 Dynamic Contract Design

To formulate the problem at the lower layer, we focus on a single cluster. For clarity and simplicity, we omit the subscript m, which denotes the cluster index, throughout this section. The contract design for other clusters can be carried out in the same manner. Note that in our setting, static contract theory is not applicable due to the incremental nature of data collection in RL. To address this, we design a separate contract for each round l. A feasible contract must satisfy both the individual rationality (IR) and incentive compatibility (IC) constraints.

Definition 1: The IR constraint ensures that each type-i agent has a non-negative utility in each round l, making participation beneficial. This means that for a type-i agent choosing contract c_i^l, we have

$$u_i^l(c_i^l, \theta_i) \geq 0, \quad \forall i \in \{1, \dots, I\}. \tag{12}$$

Definition 2: The IC constraint ensures that agents act truthfully, revealing their true type and preferring the contract designed for them over any other. This means that for a type-i agent choosing contract c_i^l maximizes their utility, which can be expressed as

$$u_i^l(c_i^l, \theta_i) \geq u_i^l(c_j^l, \theta_i), \quad \forall i, j \in \{1, \dots, I\}, \, i \neq j. \tag{13}$$

Our objective is to maximize the utility of the local server, and thus an optimization problem for contract design can be formulated as follows

$$\begin{aligned}
&\textbf{\textit{P1:}} && \underset{c_i^l}{\text{maximize}} \quad U_m \\
&\text{subject to} && (2), (4), (12), (13), \\
&p_i N R_i^l \leq B_i^l, && \forall i \in \{1, \dots, I\}, && (14a) \\
&q_i^l \leq q_{\max}^l, && \forall i \in \{1, \dots, I\}. && (14b)
\end{aligned}$$

The constraint (14a) ensures that the total reward distributed to type-i agents in round l does not exceed the allocated budget for that type. The constraint (14b) imposes an upper bound on the amount of data a type-i agent can contribute in round l, reflecting the practical limitations of RL.

To solve **P1**, we first derive the necessary conditions for the contract and simplify the IR and IC constraints. Simplifying these constraints is a common practice in contract theory to streamline the optimization process by reducing the number of binding constraints, thereby facilitating the derivation of optimal contracts.

Lemma 1 (Monotonicity): For any feasible contract, if $\theta_i \leq \theta_j$, then $q_i^l \geq q_j^l$.

Proof: Any feasible contract must satisfy the IC constraints. Thus, for any agent types i and j, we must have

$$R_i^l - \theta_i q_i^l \geq R_j^l - \theta_i q_j^l, \tag{15}$$

and

$$R_j^l - \theta_j q_j^l \geq R_i^l - \theta_j q_i^l. \tag{16}$$

From (15) and (16), we have

$$(\theta_j - \theta_i)\left(q_i^l - q_j^l\right) \geq 0. \tag{17}$$

Since $\theta_j - \theta_i \geq 0$, it follows that $q_i^l \geq q_j^l$. ■

Lemma 1 demonstrates that an agent with higher computation costs contributes less data.

Lemma 2: For any feasible contract, $q_i^l \leq q_j^l$ if and only if $R_i^l \leq R_j^l$.

Proof: Any feasible contract must satisfy the IC constraints. For type-j agent, we have

$$R_j^l - \theta_j q_j^l \geq R_i^l - \theta_j q_i^l. \tag{18}$$

Simplifying this inequality gives

$$R_j^l - R_i^l \geq \theta_j\left(q_j^l - q_i^l\right). \tag{19}$$

Necessity: Assume $q_i^l \leq q_j^l$. Since $\theta_j > 0$, we have $\theta_j\left(q_j^l - q_i^l\right) \geq 0$. Substituting it into inequality (19), we get $R_j^l - R_i^l \geq 0$, which implies $R_i^l \leq R_j^l$.

Now, for type-i agent, the IC constraint is

$$R_i^l - \theta_i q_i^l \geq R_j^l - \theta_i q_j^l \tag{20}$$

$$\Rightarrow \quad \theta_i(q_j^l - q_i^l) \geq R_j^l - R_i^l. \tag{21}$$

Sufficiency: Assume $R_i^l \leq R_j^l$. Since $\theta_i > 0$, and $\theta_i\left(q_j^l - q_i^l\right) \geq 0$, to make such inequality always hold, we must have $q_i^l \leq q_j^l$. ■

Lemma 2 means that agents who contribute more data should receive a higher reward, and vice versa.

Lemma 3: If the IR constraint is satisfied for the agent with the minimum utility, which in this case is the type I, then it is automatically satisfied for all agent types, i.e, we have

$$\begin{aligned} u_i^l(c_i^l, \theta_i) = R_i^l - \theta_i q_i^l \geq R_I^l - \theta_i q_I^l \\ \geq R_I^l - \theta_I q_I^l = u_I^l(c_I^l, \theta_I) \geq 0. \end{aligned} \tag{22}$$

Proof: From Lemma 1, we have known that an agent with a higher computation cost contributes less data. Furthermore, from Lemma 2, an agent contributing less data receives a lower reward. Since the highest computation cost corresponds to the type-I agent, it contributes the least data and receives the smallest reward.

As a result, the utility of the type-I agent is the minimum among all agents. If this minimum utility is non-negative ($u_I^l \geq 0$), it follows that all other agents, who have higher utilities, will also satisfy their IR constraints. Thus, ensuring the IR constraint for the type-I agent is sufficient to guarantee IR for all agents. ■

Lemma 4: If the IC constraint is satisfied for a type-i agent, then it also holds for a type-$(i-1)$ agent.

Proof: From the IC constraint for a type-i agent, we have

$$R_i^l - \theta_i q_i^l \geq R_{i+1}^l - \theta_i q_{i+1}^l \tag{23}$$

$$\Rightarrow \quad R_i^l \geq R_{i+1}^l - \theta_i \left(q_{i+1}^l - q_i^l \right). \tag{24}$$

Similarly, from the IC constraint for a type-$(i+1)$ agent, we have

$$R_{i+1}^l - \theta_{i+1} q_{i+1}^l \geq R_i^l - \theta_{i+1} q_i^l \tag{25}$$

$$\Rightarrow \quad R_i^l \leq R_{i+1}^l - \theta_{i+1} \left(q_{i+1}^l - q_i^l \right). \tag{26}$$

Combining (24) and (26), we arrive at the following conclusion

$$R_{i+1}^l - \theta_i \left(q_{i+1}^l - q_i^l \right) \leq R_i^l \leq R_{i+1}^l - \theta_{i+1} \left(q_{i+1}^l - q_i^l \right). \tag{27}$$

■

Lemma 4 establishes an important property of the rewards R_i^l: the reward assigned to a type-i agent must lie within a range determined by the neighboring type-$(i+1)$. The IC constraint ensures that each agent prefers the contract intended for their own type over those designed for other types. Specifically, a type-i agent should not prefer the contract for type-$(i+1)$, and vice versa. This mutual preference between adjacent types enables the global IC constraint to be reduced to a series of pairwise comparisons, commonly referred to as local downward IC constraints (LDICs) [16]. By examining the relationships between neighboring types, we can enforce the IC constraint iteratively. That is, if the LDIC holds between types i and $i+1$, and also between types $i+1$ and $i+2$, then the IC constraint is implicitly satisfied for more distant type pairs (e.g., i and $i+2$). Therefore, satisfying LDICs across all adjacent type pairs is both necessary and sufficient to ensure that the global IC condition holds for all agent types.

Using Lemmas 2, 3, and 4, we now have the necessary and sufficient conditions for achieving a feasible contract design that satisfies both the IR and IC constraints, as shown in Theorem 1.

Theorem 1. *A feasible contract must satisfy the following conditions*

(1) $u_I^l(c_I^l, \theta_I) = R_I^l - \theta_I\, q_I^l(\theta_I) \geq 0,$

(2) $R_{i+1}^l - \theta_i(q_{i+1}^l - q_i^l) \leq R_i^l \leq R_{i+1}^l - \theta_{i+1}(q_{i+1}^l - q_i^l),$

(3) $R_1^l \geq \cdots \geq R_i^l \geq \cdots \geq R_I^l, \quad q_1^l \geq \cdots \geq q_i^l \geq \cdots \geq q_I^l.$

From Theorem 1, we can derive the optimal reward as summarized in Theorem 2.

Theorem 2. *Considering the monotonicity of data contributions, the optimal reward is given by*

$$R_i^{l*} = \begin{cases} \theta_i\, q_i^l, & \text{if } i = I, \\ \theta_i\, q_i^l + \Delta_i, & \text{if } i \neq I, \end{cases} \tag{28}$$

where $\Delta_i = \sum_{j=i+1}^{I} (\theta_j - \theta_{j-1})\, q_j^l$*, and* I *represents the agent type with the lowest data contribution.*

Proof: To prove that R_i^{l*} is optimal, assume there exists another reward function $R_i^{l\circ}$ that reduces the server's cost while satisfying the constraints of a feasible contract. This would imply

$$R_i^{l\circ} < R_i^{l*}. \tag{29}$$

For the IC constraint of type i, we have

$$R_{i+1}^{l\circ} \leq R_i^{l\circ} - \theta_i \left(q_i^l - q_{i+1}^l \right). \tag{30}$$

By substituting $R_i^{l\circ} < R_i^{l*}$ into (30), we obtain

$$R_{i+1}^{l\circ} < R_i^{l*} - \theta_i \left(q_i^l - q_{i+1}^l \right) = R_{i+1}^{l*}. \tag{31}$$

By recursively applying the inequality to each higher type, we obtain

$$R_I^{l\circ} < R_I^{l*} = \theta_I \, q_I^l. \tag{32}$$

This leads to a violation of the IR constraint since the utility for user type I becomes negative. Therefore, the assumption that $R_i^{l\circ}$ results in higher server utility than R_i^{l*} is invalid.

Consequently, R_i^{l*} is proven to be the optimal reward function, satisfying both the IC and IR constraints while ensuring monotonicity. ■

Considering the specific constraints of our problem and the definition of the reward function, the updated optimization problem can be formulated as follows

$$\begin{aligned} \textbf{\textit{P2:}} \quad & \underset{q_i^l}{\text{maximize}} \quad U_m \\ & \text{subject to} \quad (2),\ (4), (14a), (14b). \end{aligned} \tag{33}$$

Note that **P2** is a reformulation from **P1**, by substituting the optimal reward function R_i^{l*}, obtained from Theorem 2, into the original objective. This reformulation removes R_i^l as a decision variable and makes the problem solely relying on data contribution q_i^l. Moreover, since the optimal rewards are constructed to satisfy both the IR and IC constraints by design, those constraints are no longer included in **P2**. It is worth mentioning that the **P2** is also convex [12].

However, in **P2**, the budget B in constraint (2) is not known in advance. Thus, achieving a closed-form solution for data contribution as a function of the budget is difficult. Unfortunately, this relationship plays a critical role in the design of the upper layer.

To address this issue, we approximate the overall impact of budget on model performance using the relationship between cluster-level accuracy A_m and budget B. From (10), this relationship effectively captures the underlying budget–data dependency without needing to fit individual functions for each agent's contribution. Therefore, it offers a practical and scalable alternative.

To obtain this mapping, we simulate *P2* across a range of budget values and observe the resulting accuracy values, as shown in Fig. 4.1. The resulting curve clearly follows a logistic trend, which can be approximated by the following function

$$A(\gamma B) = \frac{\mathcal{L}}{1 + e^{-\lambda(\gamma B - \gamma^0 B^0)}}, \tag{34}$$

where $\mathcal{L}$ represents the maximum value of the curve, λ is the logistic growth rate or steepness of the curve, and $\gamma^0 B^0$ denotes the value at the sigmoid midpoint.

4.2 Stackelberg Game Design

In this section, we solve the Stackelberg game introduced earlier. Given the hierarchical structure between the main server and local servers, we aim to derive the Stackelberg equilibrium by determining the optimal strategies for all parties. The game is formulated as follows

$$\max_{T} \Pi(T), \tag{35}$$

$$\max_{\gamma_m} \pi_m(T, \gamma_m, \gamma_{-m}), \quad 1 \leq m \leq M. \tag{36}$$

To this end, we derive the Stackelberg equilibrium by analyzing the game using backward induction. First, we address the second stage, where local servers select their optimal γ based on T. Next, in the first stage, we determine the optimal T for the main server to maximize its utility.

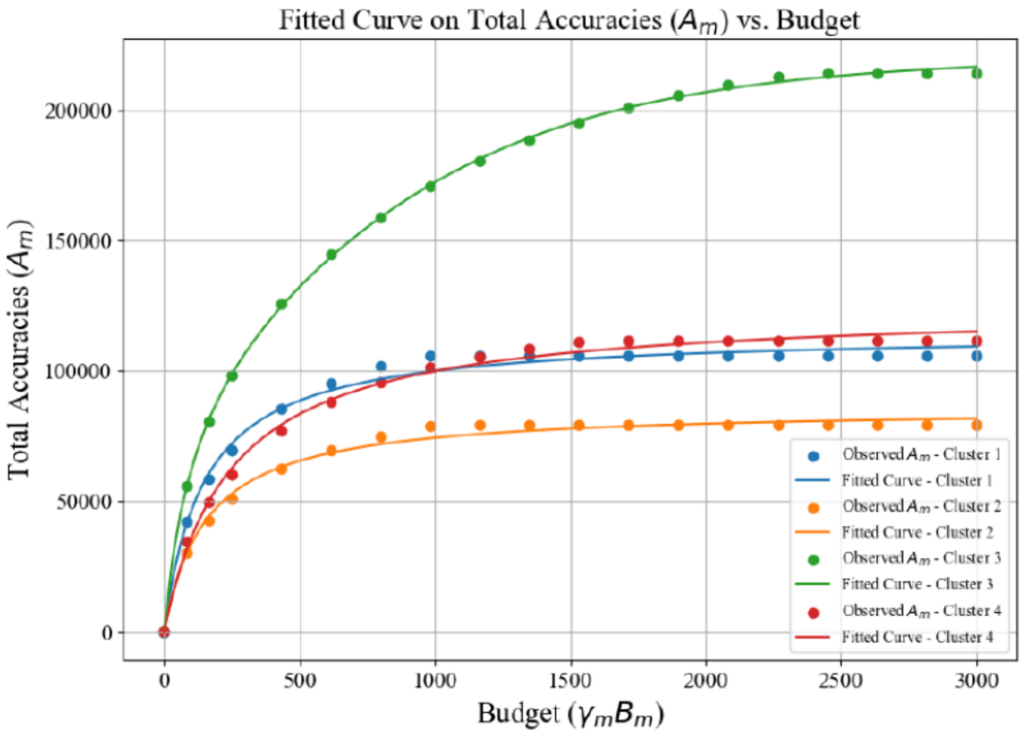


Figure 4.1: Logistic curve illustrating the relationship between total accuracy and budget data points for each cluster.

Finally, we demonstrate the existence of a unique equilibrium, ensuring that none of the parties have an incentive to deviate from their optimal strategies.

In the second stage of the game, to determine the optimal strategy for local server m, we calculate the first-order derivative of $\pi_m(\gamma_m, \gamma_{-m})$ with respect to γ_m as follows

$$\frac{\partial \pi_m(\gamma_m, \gamma_{-m})}{\partial \gamma_m} = (1 - \gamma_m)\frac{\partial B_m}{\partial \gamma_m} - B_m. \tag{37}$$

From (9), we know that B_m is a function of A_m. Therefore, the first derivation of B_m can be written as

$$\frac{\partial B_m}{\partial \gamma_m} = \frac{\frac{\partial A_m}{\partial \gamma_m} \cdot \sum_{i \neq m} A_i}{\left(\sum_{i=1}^{M} A_i\right)^2} \cdot T. \tag{38}$$

As the allocated budget for training by the local server increases, the data contribution will increase. However, since model accuracy is a concave function of data, the overall accuracy becomes a concave function of γ_m. In addition, the accuracy has a diminishing behavior with respect to both data quantity and γ_m. Therefore, the first order derivation of accuracy is non-negative with respect to γ_m, resulting in a non-negative value for the numerator and entire fraction of (38). Using this result, the second-order derivative of the local server m's utility can be expressed as

$$\frac{\partial^2 \pi_m(\gamma_m, \gamma_{-m})}{\partial \gamma_m^2} = (1-\gamma_m) \cdot \frac{\partial^2 B_m}{\partial \gamma_m^2} - 2 \cdot \frac{\partial B_m}{\partial \gamma_m}. \tag{39}$$

Using (10), we have

$$\frac{\partial^2 B_m}{\partial \gamma_m^2} = \frac{\frac{\partial^2 A_m}{\partial \gamma_m^2} \cdot \sum_{i \neq m} A_i \cdot \sum_{i=1}^{M} A_i - 2 \cdot \left(\frac{\partial A_m}{\partial \gamma_m}\right)^2}{\left(\sum_{i=1}^{M} A_i\right)^3}. \tag{40}$$

The term $\frac{\partial^2 A_m}{\partial \gamma_m^2}$ represents the second derivative of the accuracy function, A_m, with respect to γ_m. Since accuracy is inherently a concave function with respect to allocated budget for training $\gamma_m B_m$, $\frac{\partial^2 A_m}{\partial \gamma_m^2}$ is negative. In addition, because accuracy itself is non-negative, the term $\frac{\partial^2 A_m}{\partial \gamma_m^2} \cdot \sum_{i \neq m} A_i \cdot \sum_{i=1}^{M} A_i$ in (40) is also negative. Therefore, the second derivative of B_m with respect to γ_m is negative, confirming that B_m is a concave function of γ_m.

In addition, since γ_m has a value between zero and one, we have (39) to be negative. Therefore, the utility of the local server is a concave function with respect to its strategy γ_m.

For a concave utility function, the maximum value occurs at the critical point where the first derivative equals zero. Therefore, the optimal γ_m can be obtained by solving

$$\frac{\partial \pi_m(T, \gamma_m, \gamma_{-m})}{\partial \gamma_m} = 0. \tag{41}$$

By substituting (9), (37), and (38) into (41), we obtain the following equation for each local server

$$A_m^2 + A_m \cdot \sum_{i \neq m} A_i - \frac{\partial A_m}{\partial \gamma_m} \cdot (1-\gamma_m) \cdot \sum_{i \neq m} A_i = 0. \tag{42}$$

By solving the differential equation in (42), as detailed in the appendix, we derive the relationship between the accuracy A_m and the parameter γ_m for the local server m. This results in the following closed-form expression for A_m,

$$A_m = \frac{\sum_{i \neq m} A_i \cdot e^{c \sum_{i \neq m} A_i}}{(1-\gamma_m) - e^{c \sum_{i \neq m} A_i}} \tag{43}$$

We now have M equations from (43), along with another M equations estimating the accuracy A_m as a function of the allocated budget $\gamma_m B_m$, given in (34). Together, these $2M$ equations form a set of nonlinear equations, is subject to constraints imposed by the nature of the problem: accuracy values must be non-negative, and each γ_m must lie within the interval $(0,1)$. To determine the optimal values γ_m^* under these conditions, we solve the system numerically using the `least_squares` method from the `scipy.optimize` module in Python [93]. The values of A_m and γ_m^* depend on the total budget T, since γ_m^* is derived from A_m, which is itself computed from B_m, a proportion of T. Therefore, γ_m^* can only be determined once T is specified.

Next, we explain how the server determines the optimal value of the total budget. To achieve this, we first prove the existence of a unique Stackelberg equilibrium and then demonstrate the process by which the server identifies the optimal T.

Definition 3: A Nash Equilibrium (NE) is achieved with the optimal parameters $\langle \gamma_1^*, \ldots, \gamma_m^*, \ldots, \gamma_M^* \rangle$ if, for any local server m and $\gamma_m^* \geq 0$, the following condition holds

$$\pi_m(T, \gamma_m^*, \gamma_{-m}^*) \geq \pi_m(T, \gamma_m, \gamma_{-m}^*), \quad \forall \gamma_m. \tag{44}$$

Theorem 3. *A unique Stackelberg equilibrium exists for the described game between the main server and the local servers.*

Proof: The second stage of the game, where local servers aim to maximize their share of the budget and their utility, can be modeled as a non-cooperative game. In this setting, there are M local servers, which is a finite number. The strategy space for each server, defined by the parameter γ, is constrained within the closed interval $[0,1]$. Furthermore, as discussed earlier, the utility functions of the local servers are concave. According to [94], we can conclude that γ_m^* for each server m constitutes a Nash Equilibrium (NE).

Next, we demonstrate that the optimal parameters γ_m^* $(1 \leq m \leq M)$ derived in the second stage also form a Nash Equilibrium in the first stage of the game. With this, the entire two-stage Stackelberg game will establish a Stackelberg equilibrium.

First, we substitute all the optimal values γ_m^* $(1 \leq m \leq M)$ into (8), resulting in the following

$$\Pi(T) = \xi \ln\left(1 + \sum_{m=1}^{M} A_m(\gamma_m^* B_m)\right) - T. \tag{45}$$

The first-order and the second-order derivatives of $\Pi(T)$ with respect to T can be calculated as

$$\frac{\partial \Pi(T)}{\partial T} = \frac{\xi \sum_{m=1}^{M} \frac{\partial A_m}{\partial \gamma_m^*} \cdot \frac{\partial \gamma_m^*}{\partial T}}{1 + \sum_{m=1}^{M} A_m} - 1, \tag{46}$$

$$\begin{aligned} \frac{\partial^2 \Pi(T)}{\partial T^2} = \xi \Bigg(& \frac{\sum_{m=1}^{M} \frac{\partial^2 A_m}{\partial \gamma_m^{*2}} \cdot \left(\frac{\partial \gamma_m^*}{\partial T}\right)^2 + \frac{\partial A_m}{\partial \gamma_m^*} \cdot \frac{\partial^2 \gamma_m^*}{\partial T^2}}{1 + \sum_{m=1}^{M} A_m} \\ & - \frac{\left(\sum_{m=1}^{M} \frac{\partial A_m}{\partial \gamma_m^*} \cdot \frac{\partial \gamma_m^*}{\partial T}\right)^2}{\left(1 + \sum_{m=1}^{M} A_m\right)^2} \Bigg). \end{aligned} \tag{47}$$

Since, A_m is modeled by a logarithmic function, we have $\frac{\partial^2 A_m(\gamma_m^*)}{\partial \gamma_m^{*2}} < 0$. Additionally, as T increases, the local servers will reduce the percentage of the budget given to agents, because the accuracy saturates as some point. Further increasing the budget from the main server, will not result in model accuracy increment. Therefore, around optimal T, γ_m is a concave function with respect to T. By taking all these factors into consideration, the $\sum_{m=1}^{M} \left(\frac{\partial^2 A_m}{\partial \gamma_m^{*2}} \left(\frac{\partial \gamma_m^*}{\partial T}\right)^2 + \frac{\partial A_m}{\partial \gamma_m^*} \cdot \frac{\partial^2 \gamma_m^*}{\partial T^2}\right)$ is negative, which in turn makes whole (47) negative. This confirms the concavity of the main server's utility. In other words, there exists a unique optimal total budget solution for $\frac{\partial \Pi(T)}{\partial T} = 0$, denoted as T^*.

Once γ_m^* is determined and substituted into the equation $\frac{\partial \Pi(T)}{\partial T} = 0$, the optimal value of the total budget T^* can be derived.

The existence of a Stackelberg equilibrium is guaranteed for the two-stage Stackelberg game, as both the main server and local servers are capable of independently optimizing their strategies to maximize their own utilities [95]. ■

Algorithm 1 Federated Reinforcement Learning with Hierarchical Incentives (FRHI)

```
1:  for each iteration $t$ do
2:    Initialization:
3:    Set the number of clusters $M$, agents per cluster $N_m$, agent type distribution $p_{m,i}$, agent types
      $I_m$, and maximum data contribution $q^l_{m,\max}$.
4:    Contract Theory (Lower Layer):
5:    for each cluster $m$ do
6:      Simplify the IR and IC constraints using Theorem 1.
7:      Compute the optimal reward $R^{l*}_i$ as a function of $q^l_i$ using Theorem 2.
8:      Substitute $R^{l*}_i$ into problem $P2$ to reformulate it solely in terms of $q^l_i$.
9:      for Samples of $\gamma_m B_m$ do
10:       Solve $P2$ to obtain optimal $q^l_i$ for different $\gamma_m B_m$ values.
11:     end for
12:     Substitute $q^l_i$ into (10) to derive $A_m$.
13:     Fit a logistic function to the obtained values to estimate $A_m$ as a function of $\gamma_m B_m$.
14:     Substitute (9) into the estimated function to express $A_m$ as a function of $\gamma_m$ and $T$.
15:   end for
16:   Stackelberg Game (Upper Layer):
17:   Solve (41) to obtain (43).
18:   Derive $A_m$ and $\gamma_m$ as functions of $T$ using estimated function at step 14 and (43).
19:   Substitute $A_m$ and $\gamma^*_m$ into (45) and (46), and solve for $T^*$.
20:   Compute $A_m$ and $\gamma^*_m$ using $T^*$.
21:   For each local server, calculate $B_m$ using (9).
22:   for each cluster $m$ do
23:     Compute $q^l_i$ by solving problem $P2$ using the value of $\gamma^*_m B_m$ as the allocated budget.
24:   end for
25:   Check the termination condition. If satisfied, stop the process.
26: end for
```

We now have all the necessary components to complete the design of the proposed framework. The detailed procedure is outlined in Algorithm 1, referred to as the Federated Reinforcement Learning with Hierarchical Incentives (FRHI).

Chapter 5

Simulation Results

In this chapter, we evaluate the effectiveness and practicality of the proposed FRHI framework through extensive simulations. The aim is to validate both the analytical properties and the performance gains of our incentive mechanism under realistic system settings. We first outline the simulation setup, including the agent and cluster configurations, parameter distributions, and budget constraints.

Subsequently, we analyze the behavior of the local and global components of the system by examining metrics such as server utility, agent data contribution, and adherence to IR and IC constraints. We also compare our method against three benchmark strategies to highlight the performance benefits and budget efficiency introduced by our hierarchical design. The simulation results not only confirm the theoretical soundness of our model but also demonstrate its superiority in optimizing resource allocation and incentive effectiveness.

5.1 Experiment Setup

In the simulations, we considered four different clusters, each containing 20, 15, 30, and 25 agents, respectively. The number of agent types is considered to be 5, 4, 6, and 4 for each cluster, respectively. The computation cost, denoted as θ, is randomly generated within the range $[0.1, 0.9]$. The probability of each agent being classified as a specific type is also generated randomly.

The maximum allowed contribution, $q_{\max}$, is represented as a matrix of size $I \times L$, where each

Table 5.1: Evaluation Parameters

Parameter	**Value**
M	4 (Number of clusters)
N_m	$N_1 = 20$, $N_2 = 15$, $N_3 = 30$, $N_4 = 25$
I_m	$I_1 = 5$, $I_2 = 4$, $I_3 = 6$, $I_4 = 4$
θ	[0.1, 0.9]
q_{max}	[1, 35]
σ	[350, 500]
η	1
ζ	800

element indicates $q_{\max}$ for a specific agent type at each round. Additionally, $q_{\max}$ increases over rounds. In our setting, $q_{\max}$ generally ranges between $[1, 35]$.

Finally, σ is generated within the range $[350, 500]$, and η is considered to be 1 for each cluster. ζ for the main server is set to a default value of 800. Table 5.1 provides a summary of the parameter settings.

Furthermore, to demonstrate the effectiveness of our approach, we simulate the following methods for comparison:

- **Federated Learning with Constant Allowance (FLCA):** A standard federated learning setting in which $q_{\max}$ remains constant across all rounds and agent types. The budget allocation across local servers and agents follows the same scheme as in our proposed method.

- **Federated Reinforcement Learning with Uniform Allocation (FRUA):** The total budget T is evenly allocated among local servers. Each local server then distributes its budget equally across all rounds and among agent types.

- **Federated Reinforcement Learning with Linear Allocation (FRLA):** The total budget T is evenly distributed among local servers. Each local server assigns an equal budget to all agent types within a given round, but the allocated budget per round increases linearly over time.

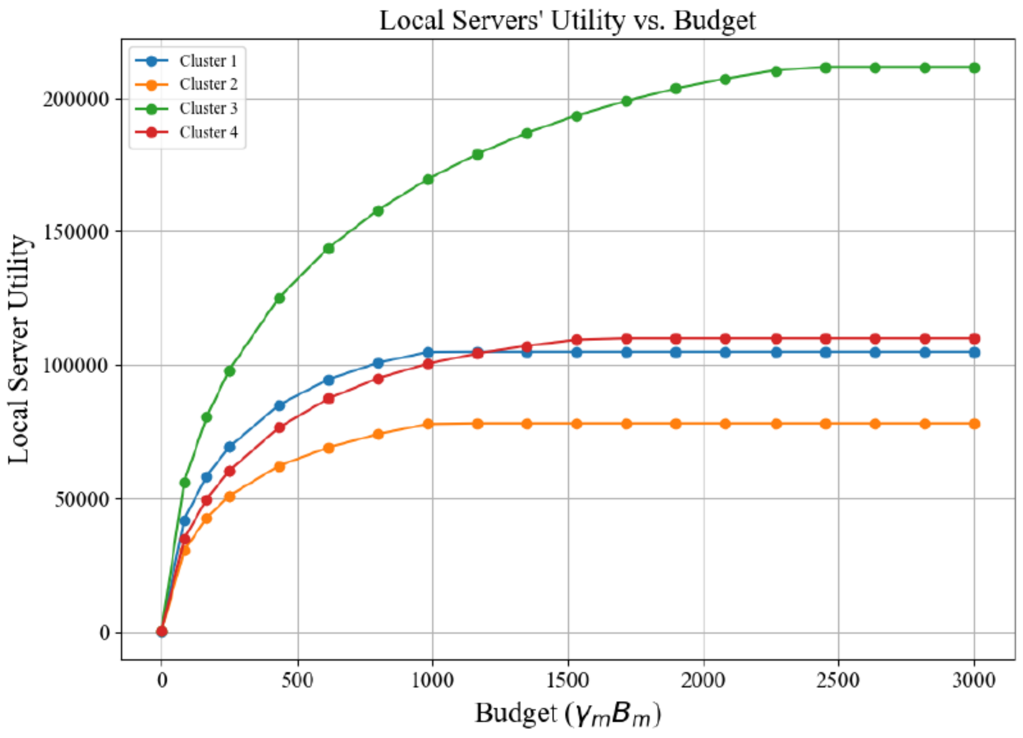
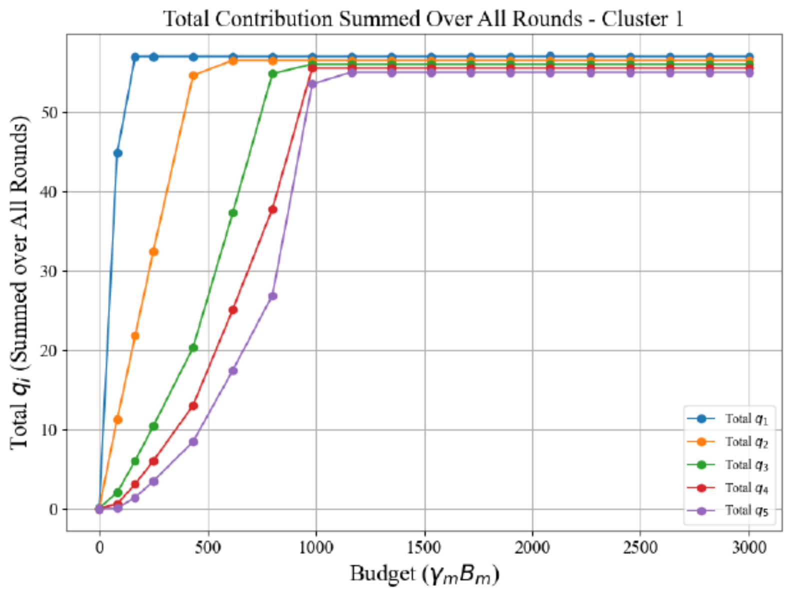


Figure 5.1: Utility of local servers as a function of budget allocation.

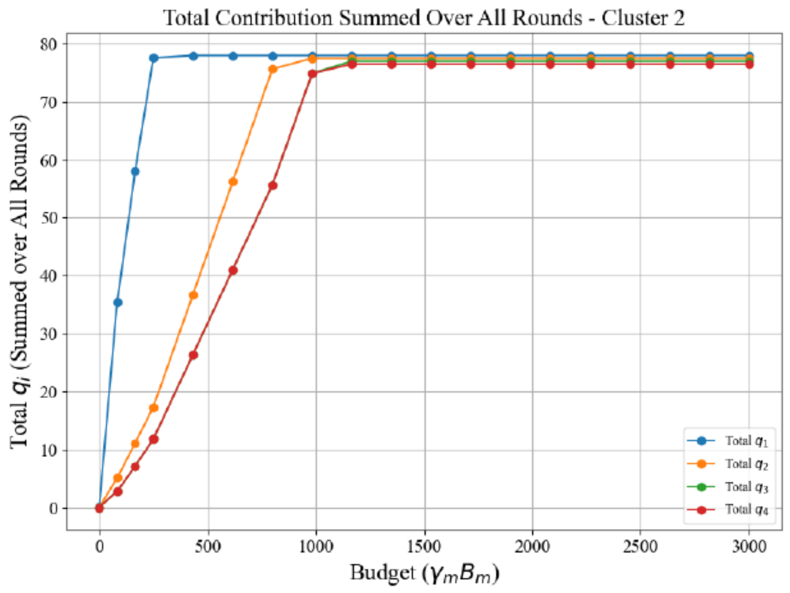
5.2 Experiment Results

We first verify the effect of the allocated budget on the utility of local servers and agent data contribution in each cluster. Accordingly, in Figs. 5.1, 5.3 and 5.3, we present the trend of local server utility and agent data contribution in contract theory as a function of budget variation within the range $[0, 3000]$.

As observed in Fig. 5.1, the utility of local servers begins to saturate beyond a certain budget threshold. This observation indicates that after a certain point, increasing the allocated budget does not necessarily lead to performance improvements. This could be due to the fact that, beyond a certain limit, there may no longer be significant new features to be learned from the environment, and agents may not be able to contribute additional information. As shown in Figs. 5.3 and 5.3, the total contributed data stabilizes beyond a certain budget threshold. Thus, the accuracy of the model remains unchanged, leading to a constant utility for the local server.

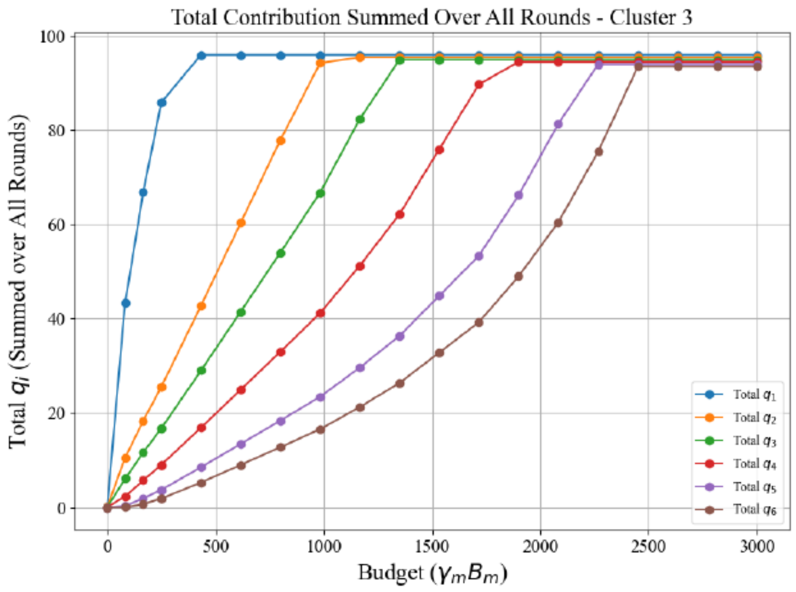


(a) Cluster 1

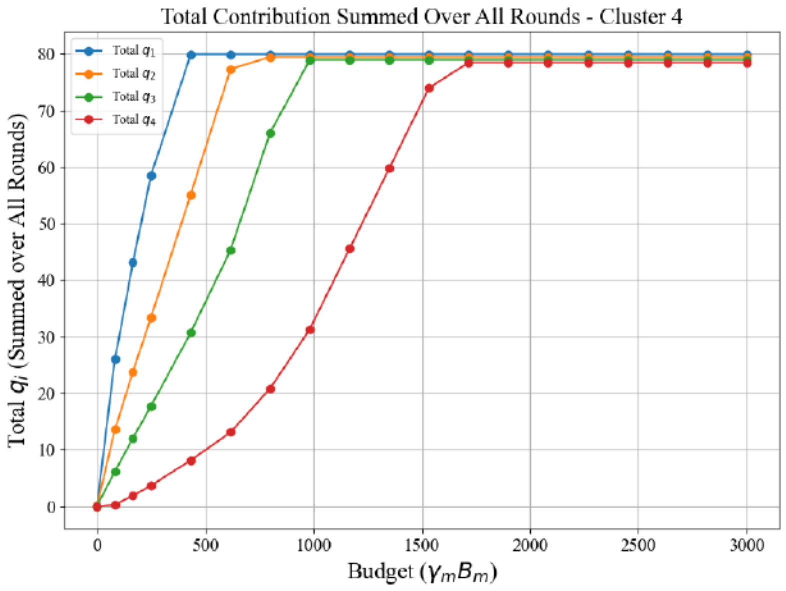


(b) Cluster 2

Figure 5.2: Data contribution vs. budget for Clusters 1 and 2.



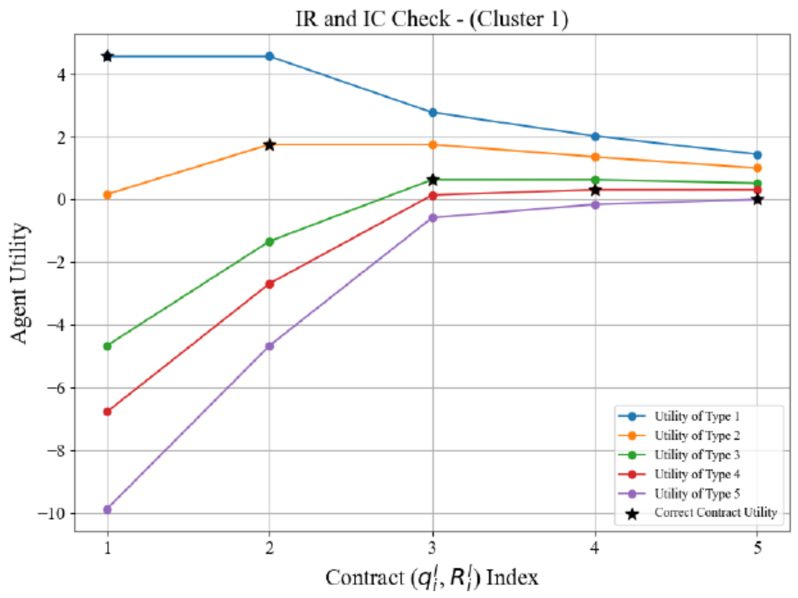
(a) Cluster 3



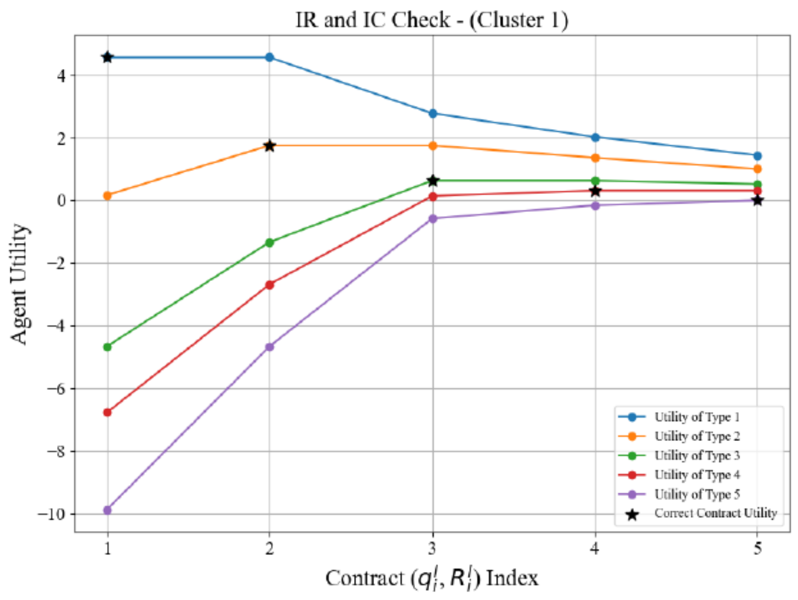
(b) Cluster 4

Figure 5.3: Data contribution vs. budget for Clusters 3 and 4.

Next, we verify that the q values obtained using the optimal T satisfy both the IR and IC constraints. We calculate agent utility using (5). For each agent, we compute its utility by evaluating all available contracts.



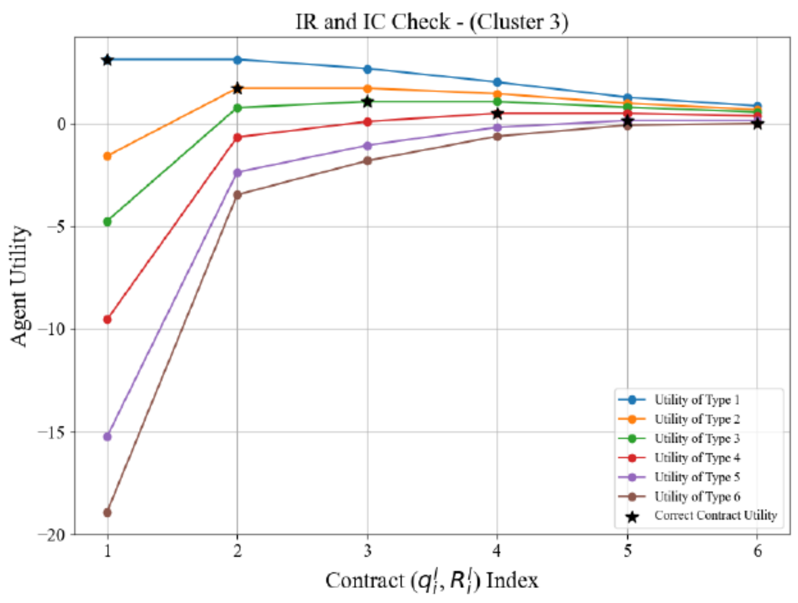
(a) Cluster 1



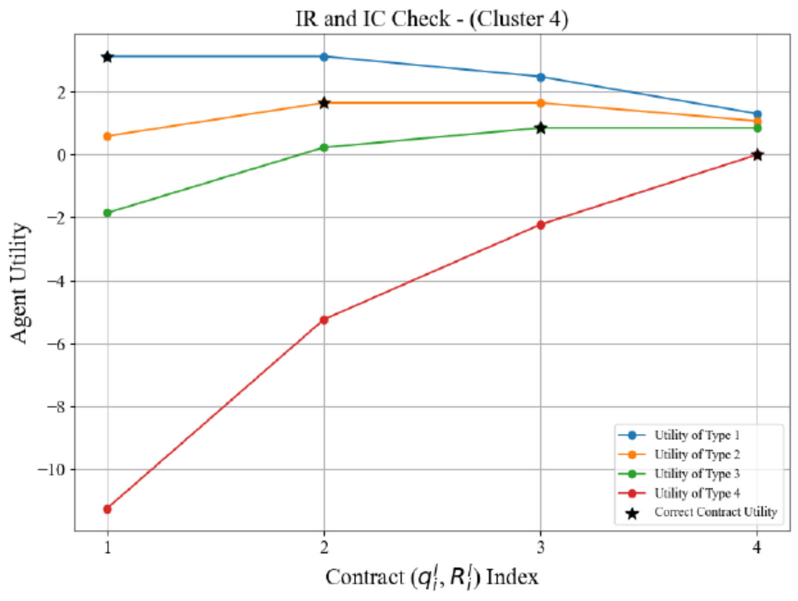
(b) Cluster 2

Figure 5.4: IR and IC constraint satisfaction for Clusters 1 and 2 in the last round of training.

Figs. 5.4 and 5.5 presents the utility of each agent type in all four clusters when choosing contracts designed for different types. The x-axis represents the contract type selected, and the y-axis shows the resulting utility. Each colored line corresponds to a specific agent type, while the black star markers indicate the utility received when the agent chooses its designated (truthful) contract.



(a) Cluster 3



(b) Cluster 4

Figure 5.5: IR and IC constraint satisfaction for Clusters 3 and 4 in the last round of training.

Table 5.2: Comparison of $q_{\max}$, q, Reward, and θ for the Last Round Across Clusters

Cluster	**User Type**	θ	$q_{\max}$	q	**Reward**
1	1	0.16	19.80	19.80	7.70
	2	0.38	19.7	12.66	6.57
	3	0.62	19.6	4.61	3.51
	4	0.73	19.5	2.99	2.50
	5	0.89	19.4	1.97	1.75
2	1	0.24	27.20	27.20	10.93
	2	0.47	27.10	8.45	6.41
	3	0.75	27.00	6.11	5.29
	4	0.86	26.90	6.11	5.29
3	1	0.13	32.40	31.23	7.23
	2	0.28	32.30	9.35	4.36
	3	0.38	32.20	6.37	3.51
	4	0.54	32.10	3.74	2.51
	5	0.72	32.00	1.92	1.53
	6	0.84	31.90	1.22	1.02
4	1	0.21	30.00	30.00	8.33
	2	0.32	29.90	9.35	6.15
	3	0.42	29.80	6.37	4.18
	4	0.81	29.70	1.22	1.77

As observed in all subfigures, each agent type obtains the highest utility when selecting the contract intended for its own type. This confirms that the IC constraint holds. Moreover, the utility values at the truthful contracts are consistently positive, satisfying the IR constraint. Similar trends are observed in earlier rounds, where utility values increase progressively, reflecting improved learning and convergence of the contract mechanism.

Fig. 5.6 illustrates the main server's utility over a budget range of $T \in [100, 9000]$, revealing a clear concave trend. To identify the optimal value of T, we employed the `minimize_scalar` function from `scipy.optimize`, which minimizes the negative utility function $-\Pi(T)$, thereby maximizing $\Pi(T)$. The optimal point, as marked in the figure, is found to be $(2257.58, 6343.66)$.

After we have derived the optimal T, we can determine the optimal strategies for local servers, including budget allocation for training in each cluster. Table 5.2 summarizes the results from the final training round in the lower layer, demonstrating the numerical validity of our framework.

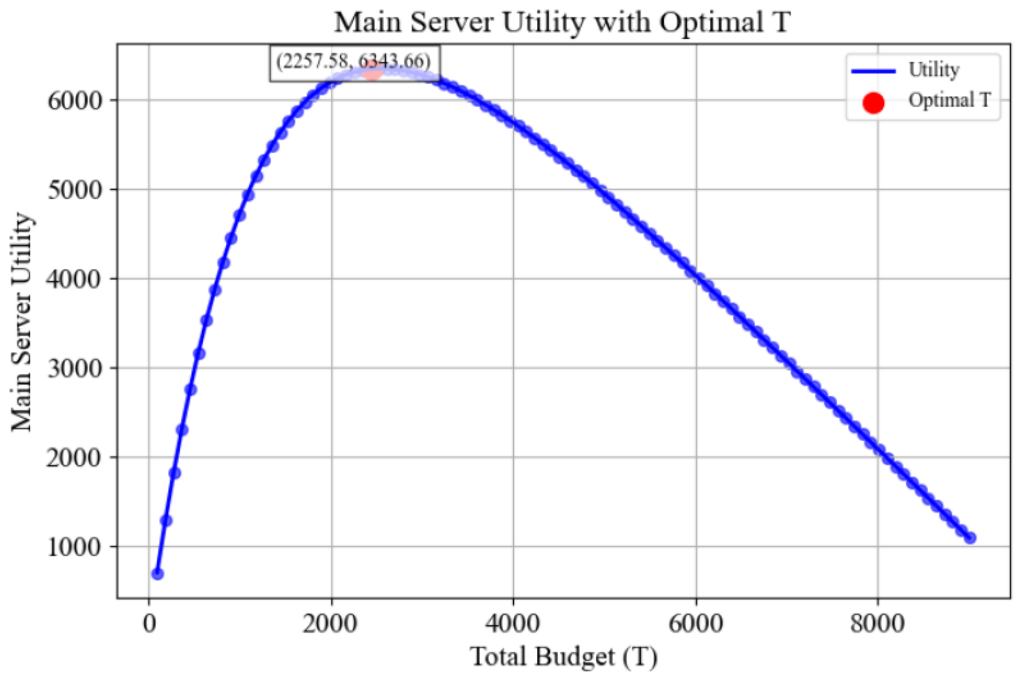


Figure 5.6: Main server utility as a function of total budget T.

The applied bounds on q ensure agents' data contributions remain within permissible limits, as confirmed by comparisons with $q_{\max}$. The reported rewards r_i^l for each agent type i fully cover the corresponding computation costs, validating the effectiveness of the incentive design.

In Fig. 5.7, we present a bar chart comparison of the optimal T values that maximize the main server's utility. As illustrated, our proposed approach (FRHI) requires the smallest value of T while achieving higher utility for the main server compared to both FRUA and FRLA. Specifically, FRUA and FRLA require 59.95% and 49.35% more budget than FRHI, yet result in 3.44% and 1.78% lower utility, respectively.

The FRUA and FRLA do not consider performance-based budget allocation. To maximize profit, these methods distribute the budget uniformly, paying the same amount to all local servers regardless of their performance. This results in higher payments but lower overall profit for the main server.

FLCA consumes 7.21% less budget compared to FRHI because $q_{\max}$ remains fixed across

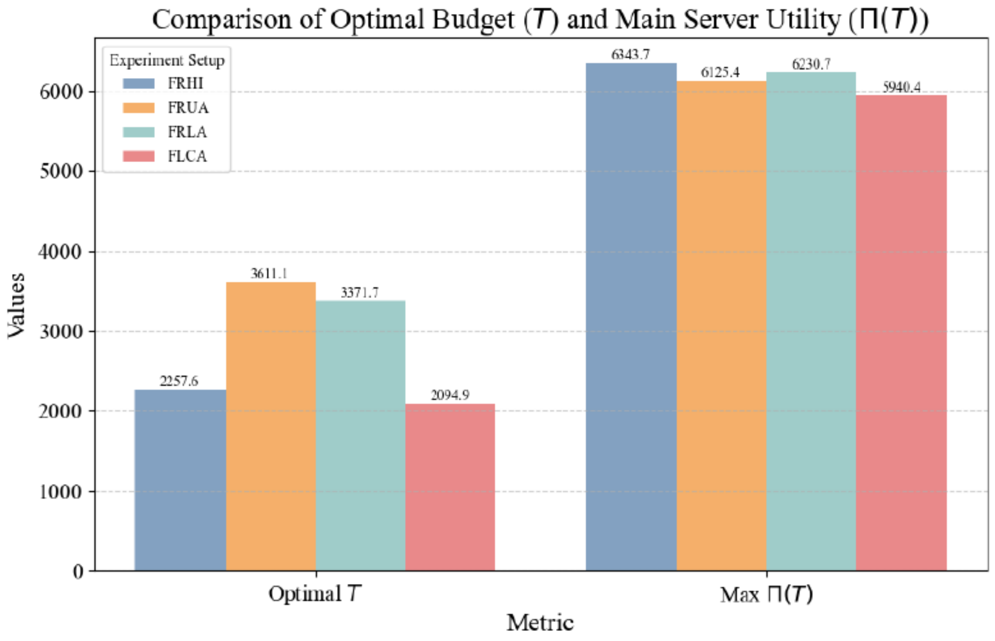


Figure 5.7: Comparison of main server utility and total budget usage across models

rounds, limiting the growth of data contribution q over rounds and thereby reducing the overall budget required. However, this constraint results in a degradation of utility. As illustrated in Fig. 5.7, FRHI achieves a 6.36% higher utility, demonstrating the advantage of allowing data contributions to increase across rounds. This comparison highlights the fundamental distinction between FL and FRL, and underscores the need for incentive mechanisms that explicitly account for key characteristics discussed in this paper, such as the incremental nature of agent data and the presence of budget-constrained servers.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis by summarizing the key contributions of the proposed hierarchical incentive mechanism for FRL and outlining promising directions for future research. The proposed framework addresses critical challenges in budget-constrained and heterogeneous FRL environments, offering both theoretical insights and practical benefits. We reflect on its overall impact and highlight opportunities for extending its capabilities.

6.1 Conclusions

In this thesis, we proposed a hierarchical incentive mechanism tailored for FRL environments characterized by constrained resources and heterogeneous agents. The motivation behind this work stems from the need to balance three competing goals: maximizing the utility of the main server, incentivizing local agent participation, and preserving overall model performance in a scalable and practical system.

To address these challenges, we introduced a two-layer incentive framework. At the lower layer, contract theory was employed to design reward mechanisms that consider agent heterogeneity and asymmetric information. These contracts ensure both individual rationality and incentive compatibility. As a result, agents are fairly compensated and motivated to participate truthfully, even under private information about their computation costs. At the upper layer, we modeled the budget allocation problem between the main server and local servers as a Stackelberg game, allowing for optimal distribution of limited resources while maximizing system-wide utility.

To manage environment heterogeneity and improve scalability, the system adopts a clustered structure. Clients are grouped based on environmental similarities, and each cluster operates semi-independently with its own local server. This design allows for parallelized training and a reduced coordination burden on the main server.

We conducted a rigorous theoretical analysis of the incentive mechanisms, showing that the contract rewards and budget allocations satisfy necessary constraints while optimizing utilities. The simulation results further validated the effectiveness of our approach. Specifically, our method achieved higher utility for both local and main servers compared to several baseline methods, while maintaining feasible reward distribution and respecting data contribution limits. Additionally, the simulations revealed that increasing the training budget beyond a certain threshold leads to performance saturation, emphasizing the importance of cost-effective incentive strategies.

In conclusion, our hierarchical incentive design introduces a flexible and effective solution to key FRL challenges. It combines game-theoretic and economic tools to enhance collaboration in decentralized learning environments, ensuring that participants are adequately incentivized and resources are judiciously allocated.

6.2 Future Work

While our proposed model offers a robust foundation for incentive-compatible FRL, several key challenges remain unaddressed. Future research can extend this work by focusing on the following areas:

1. Adaptive and Real-Time Incentive Mechanisms

Current contract and Stackelberg models assume static computation costs and fixed agent behavior during each round. However, in practical FRL systems, agent availability, computation capacity, and data quality can change over time. Future research could explore adaptive contract mechanisms that dynamically adjust reward offers based on real-time agent feedback, learning progress, or behavioral trends. Real-time optimization algorithms, such as online convex programming or reinforcement-based pricing strategies, could be integrated to adjust incentives on-the-fly and handle streaming agent participation.

2. Scalability with Large-Scale Networks

As the number of clusters and agents increases, the overhead of managing contracts, reward calculations, and budget allocation also grows. To maintain scalability, future work could explore decentralized versions of the current framework, where clusters operate autonomously with limited supervision from the main server. Hierarchical aggregation strategies or meta-learning approaches could be used to generalize reward structures across clusters. Additionally, techniques such as model pruning or client sampling can reduce the number of active agents per round, optimizing computational resources while preserving model performance.

3. Communication Cost and Overhead Reduction

A critical barrier to real-world deployment of FRL systems is the cost of communication between agents and local servers. While this work focuses on computation incentives, future work should integrate communication cost as part of the utility functions in both contract and Stackelberg formulations. Agents with high communication latency or limited bandwidth should receive tailored contracts that balance data contribution with minimal overhead. Budget-aware communication protocols such as gradient compression, quantization, or adaptive upload frequency could be incorporated to minimize transmission cost.

4. Robust Communication in Unstable or Constrained Environments

In many practical applications, especially in mobile or remote settings, reliable communication between agents and local servers cannot be guaranteed. Future research should explore incentive-compatible mechanisms that are robust to partial connectivity, intermittent dropout, or delayed updates. For instance, techniques from delay-tolerant networking (DTN) or federated dropout resilience can be adapted to ensure learning continues even when some agents are offline temporarily. Local caching, asynchronous aggregation, or edge computing can further mitigate the effects of unstable communication.

5. Privacy and Security Considerations

To ensure wide adoption of FRL in privacy-sensitive applications, future research could explore the integration of privacy-preserving technologies such as differential privacy, homomorphic encryption, or secure multi-party computation into the incentive design. The challenge lies in designing contracts and reward mechanisms that are compatible with encrypted or obfuscated contributions, while maintaining fairness and effectiveness. Ensuring incentive compatibility in the presence of privacy constraints will be a critical direction for future study.

6. Fairness and Long-Term Participation

While this work ensures fairness through IR and IC constraints, additional fairness considerations such as equal opportunity, contribution-weighted rewards, or long-term engagement incentives remain open problems. Future work can explore how to ensure that historically underperforming or disadvantaged agents are not excluded from participation. Longitudinal incentive models could consider agent loyalty, consistency, or collaborative behavior in computing future rewards.

In summary, this thesis presents a robust and flexible incentive framework for FRL, integrating economic theory with distributed learning to address real-world deployment challenges. By laying a foundation for adaptive, scalable, and fair agent collaboration, this work opens up promising directions for both theoretical advancement and practical implementation in future decentralized AI systems.

Appendix A

Appendix

In this section, we present the complete step-by-step procedure to solve the differential equation discussed in the paper. The equations are presented in the correct sequence with their detailed derivations.

We start with the following quadratic equation:

$$A_m^2 + \left(\sum_{j\neq m} A_j\right) A_m - \left(\sum_{j\neq m} A_j\right) A'_m + \left(\sum_{j\neq m} A_j\right) \gamma_m A'_m = 0. \tag{48}$$

Rearranging terms:

$$A_m^2 + \left(\sum_{j\neq m} A_j\right) A_m = A'_m \left[\sum_{j\neq m} A_j(1-\gamma_m)\right]. \tag{49}$$

From (49), we can express A'_m as:

$$A'_m = \frac{A_m^2 + \left(\sum_{j\neq m} A_j\right) A_m}{\sum_{j\neq m} A_j(1-\gamma_m)}. \tag{50}$$

Differentiating both sides, we obtain:

$$\frac{dA_m}{A_m^2 + \left(\sum_{j\neq m} A_j\right) A_m} = \frac{d\gamma_m}{\sum_{j\neq m} A_j(1-\gamma_m)}. \tag{51}$$

By simplifying, we arrive at the relationship:

$$A_m^2 + \left(\sum_{j\neq m} A_j\right) A_m = A_m \left(A_m + \sum_{j\neq m} A_j\right). \tag{52}$$

Decomposing into partial fractions:

$$\frac{1}{A_m\left(A_m + \sum_{j\neq m} A_j\right)} = \frac{A}{A_m} + \frac{B}{A_m + \sum_{j\neq m} A_j}. \tag{53}$$

Where the coefficients are given by:

$$A + B = 0, \tag{54}$$

$$A \sum_{j\neq m} A_j = 1, \tag{55}$$

$$A = \frac{1}{\sum_{j\neq m} A_j}, \quad B = -\frac{1}{\sum_{j\neq m} A_j}. \tag{56}$$

Substituting the coefficients into (53), we have:

$$\frac{1}{A_m\left(A_m + \sum_{j\neq m} A_j\right)} = \frac{1/\sum_{j\neq m} A_j}{A_m} - \frac{1/\sum_{j\neq m} A_j}{A_m + \sum_{j\neq m} A_j}. \tag{57}$$

Integrating both sides:

$$\left(\frac{1}{\sum_{j\neq m} A_j}\frac{1}{A_m} - \frac{1}{\sum_{j\neq m} A_j}\frac{1}{A_m + \sum_{j\neq m} A_j}\right) dA_m = \frac{1}{\sum_{j\neq m} A_j}\frac{1}{1-\gamma_m} d\gamma_m. \tag{58}$$

The integral on the left-hand side becomes:

$$\int \left(\frac{1}{\sum_{j\neq m} A_j}\frac{1}{A_m} - \frac{1}{\sum_{j\neq m} A_j}\frac{1}{A_m + \sum_{j\neq m} A_j}\right) dA_m = \frac{1}{\sum_{j\neq m} A_j}\ln|A_m| - \frac{1}{\sum_{j\neq m} A_j}\ln|A_m + \sum_{j\neq m} A_j|. \tag{59}$$

The integral on the right-hand side simplifies to:

$$\int \frac{1}{\sum_{j\neq m} A_j} \frac{1}{1-\gamma_m} d\gamma_m = -\frac{1}{\sum_{j\neq m} A_j} \ln|1-\gamma_m|. \tag{60}$$

Combining the results:

$$\frac{1}{\sum_{j\neq m} A_j} \ln|A_m| - \frac{1}{\sum_{j\neq m} A_j} \ln|A_m + \sum_{j\neq m} A_j| = -\frac{1}{\sum_{j\neq m} A_j} \ln|1-\gamma_m| + C. \tag{61}$$

Simplifying further:

$$\ln|A_m| - \ln|A_m + \sum_{j\neq m} A_j| = -\ln|1-\gamma_m| + \sum_{j\neq m} A_j \cdot C. \tag{62}$$

From this, we derive:

$$\ln\left(\frac{A_m}{A_m + \sum_{j\neq m} A_j}\right) = -\ln|1-\gamma_m| + \sum_{j\neq m} A_j \cdot C. \tag{63}$$

Rewriting:

$$\frac{A_m}{A_m + \sum_{j\neq m} A_j} = k \cdot \frac{1}{|1-\gamma_m|}, \tag{64}$$

where $k = e^{\sum_{j\neq m} A_j \cdot C}$.

Expressing A_m explicitly:

$$A_m = k \cdot \frac{1}{|1-\gamma_m|} \left(A_m + \sum_{j\neq m} A_j\right). \tag{65}$$

Simplifying further:

$$A_m \left(1 - \frac{k}{|1-\gamma_m|}\right) = \frac{k \sum_{j\neq m} A_j}{|1-\gamma_m|}. \tag{66}$$

Finally, solving for A_m:

$$A_m = \frac{k \sum_{j\neq m} A_j}{|1-\gamma_m| - k}.is \tag{67}$$

This concludes the derivation of A_m and its associated equations.

Bibliography

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 54. PMLR, 2017, pp. 1273–1282.

[2] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, “Federated learning for healthcare informatics,” *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[4] J. Qi, Q. Zhou, L. Lei, and K. Zheng, “Federated reinforcement learning: Techniques, applications, and open challenges,” *arXiv preprint arXiv:2108.11887*, 2021.

[5] A. M. Elbir, B. Soner, S. Çöleri, D. Gündüz, and M. Bennis, “Federated learning in vehicular networks,” in *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, 2022, pp. 72–77.

[6] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[7] H. Jin, Y. Peng, W. Yang, S. Wang, and Z. Zhang, “Federated reinforcement learning with environment heterogeneity,” in *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of

Machine Learning Research, vol. 151. PMLR, 2022, pp. 18–37. [Online]. Available: https://proceedings.mlr.press/v151/jin22a.html

[8] Z. Chen, Z. Yu, D. Zhang, and X. Zhou, "Data heterogeneity-aware federated reinforcement learning for edge computing systems," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 2189–2201, 2023.

[9] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE journal on selected areas in communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[10] H. Wang, S. He, Z. Zhang, F. Miao, and J. Anderson, "Momentum for the win: Collaborative federated reinforcement learning across heterogeneous environments," in *Proceedings of the 41st International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 2024, pp. 50 530–50 560. [Online]. Available: https://proceedings.mlr.press/v235/wang24v.html

[11] Y. Xie, C. Zhang, H. Bai, B. Yu, W. Li, and Y. Gao, "Client-level data heterogeneity in federated reinforcement learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121000381

[12] M. Akbari, J. Cai, and G. Li, "Incentive mechanism design in federated reinforcement learning with uniform and non-uniform data bounds," TechRxiv, Apr. 2025, preprint. [Online]. Available: https://doi.org/10.36227/techrxiv.174586930.02681359/v1

[13] J. Yan, T. Chen, B. Xie, Y. Sun, S. Zhou, and Z. Niu, "Hierarchical federated learning: Architecture, challenges, and its implementation in vehicular networks," *ZTE Communications*, vol. 21, no. 1, p. 38, 2023.

[14] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Federated deep reinforcement learning for resource allocation in o-ran slicing," *arXiv preprint arXiv:2208.01736*, 2022.

[15] A. Shamsian, G. Elidan, A. Navon, and G. Chechik, "Personalized federated learning using the shapley value," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9489–9499.

[16] W. Y. B. Lim, S. Garg, Z. Xiong, D. Niyato, C. Leung, C. Miao, and M. Guizani, "Dynamic contract design for federated learning in smart healthcare applications," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 853–16 862, 2020.

[17] Y. Zhan, P. Li, S. Guo, and Z. Qu, "Incentive mechanism design for federated learning: Challenges and opportunities," *IEEE Network*, vol. 35, no. 4, pp. 310–317, 2021.

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[19] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[20] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[21] C. Szepesvári, *Algorithms for reinforcement learning*. Springer nature, 2022.

[22] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[23] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[24] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," *IEEE access*, vol. 8, pp. 209 320–209 344, 2020.

[25] W. Qiang and Z. Zhongli, "Reinforcement learning model, algorithms and its application," in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. IEEE, 2011, pp. 1143–1146.

[26] I. Restack, "Ai for industrial automation: Reinforcement learning in industrial settings," *Restack Insights*, 2023. [Online]. Available: https://www.restack.io/p/ai-for-industrial-automation-answer-reinforcement-learning-industrial-settings-cat-ai

[27] MDPI, "Reinforcement learning for energy systems optimization," *MDPI Electronics*, vol. 13, no. 8, p. 1459, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/13/8/1459

[28] EEP, "Reinforcement learning improves smart grid management," *EEP Tech Insights*, 2025. [Online]. Available: https://eepower.com/tech-insights/reinforcement-learning-improves-smart-grid-management/

[29] Z. Ding and H. Dong, "Challenges of reinforcement learning," *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 249–272, 2020.

[30] A. Srinivasan, "Reinforcement learning: Advancements, limitations, and real-world applications," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 7, p. 08, 2023.

[31] Y. Li, "Reinforcement learning in practice: Opportunities and challenges," *arXiv preprint arXiv:2202.11296*, 2022.

[32] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[33] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, K. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1175–1191.

[34] K. J. Rahman, F. Ahmed, N. Akhter, M. Hasan, R. Amin, K. E. Aziz, A. M. Islam, M. S. H. Mukta, and A. N. Islam, "Challenges, applications and design aspects of federated learning: A survey," *IEEe Access*, vol. 9, pp. 124 682–124 700, 2021.

[35] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: challenges and applications," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.

[36] I. Schoinas, A. Triantafyllou, D. Ioannidis, D. Tzovaras, A. Drosou, K. Votis, T. Lagkas, V. Argyriou, and P. Sarigiannidis, "Federated learning: Challenges, sota, performance improvements and application domains," *IEEE Open Journal of the Communications Society*, 2024.

[37] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[38] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information processing & management*, vol. 59, no. 6, p. 103061, 2022.

[39] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[40] S. K. Lo, Q. Lu, L. Zhu, H.-Y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *Journal of Systems and Software*, vol. 191, p. 111357, 2022.

[41] H. Zhang, J. Bosch, and H. H. Olsson, "Federated learning systems: Architecture alternatives," in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2020, pp. 385–394.

[42] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8866–8870.

[43] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE, 2020, pp. 1–6.

[44] Y. Cui, K. Cao, J. Zhou, and T. Wei, "Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 550–563, 2021.

[45] J. Chen, L. Zhang *et al.*, "Hed-fl: A hierarchical, energy-efficient, and dynamic approach for edge federated learning," *Journal of Parallel and Distributed Computing*, 2023.

[46] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–9.

[47] Y. Yan, X. Tong, and S. Wang, "Clustered federated learning in heterogeneous environment," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[48] A. Taïk, Z. Mlika, and S. Cherkaoui, "Clustered vehicular federated learning: Process and optimization," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[49] E. C. Pinto Neto, S. Sadeghi, X. Zhang, and S. Dadkhah, "Federated reinforcement learning in iot: Applications, opportunities and open challenges," *Applied Sciences*, vol. 13, no. 11, p. 6497, 2023.

[50] G. Chen and Q. Wu, "A review of personalized federated reinforcement learning," *International Journal of Computer Science & Information Technology*, vol. 3, no. 1, pp. 1–14, 2024.

[51] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555–4562, 2019.

[52] G. Lan, D.-J. Han, A. Hashemi, V. Aggarwal, and C. G. Brinton, "Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis," *arXiv preprint arXiv:2404.08003*, 2024.

[53] F. X. Fan, Y. Ma, Z. Dai, C. Tan, B. K. H. Low, and R. Wattenhofer, "Fedhql: Federated heterogeneous q-learning," *arXiv preprint arXiv:2301.11135*, 2023.

[54] F. X. Fan, C. Tan, Y.-S. Ong, R. Wattenhofer, and W.-T. Ooi, “Fedrlhf: A convergence-guaranteed federated framework for privacy-preserving and personalized rlhf,” *arXiv preprint arXiv:2412.15538*, 2024.

[55] C. Zhang, H. Wang, A. Mitra, and J. Anderson, “Finite-time analysis of on-policy heterogeneous federated reinforcement learning,” *arXiv preprint arXiv:2401.15273*, 2024.

[56] N. Asadi, S. H. Hosseini, M. Imani, D. P. Aldrich, and S. F. Ghoreishi, “Privacy-preserved federated reinforcement learning for autonomy in signalized intersections,” in *International Conference on Transportation and Development 2024*, 2024, pp. 390–403.

[57] S. Trindade, L. F. Bittencourt, and N. L. da Fonseca, “Resource management at the network edge for federated learning,” *Digital Communications and Networks*, vol. 10, no. 3, pp. 765–782, 2024.

[58] Z. Ming, H. Yu, and T. Taleb, “Federated deep reinforcement learning for prediction-based network slice mobility in 6 g mobile networks,” *IEEE Transactions on Mobile Computing*, 2024.

[59] X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low, “Fault-tolerant federated reinforcement learning with theoretical guarantee,” *Advances in neural information processing systems*, vol. 34, pp. 1007–1021, 2021.

[60] R. Zeng, C. Zeng, X. Wang, B. Li, and X. Chu, “A comprehensive survey of incentive mechanism for federated learning,” *arXiv preprint arXiv:2106.15406*, 2021.

[61] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, and J. Li, “Incentive mechanisms for federated learning: From economic and game theoretic perspective,” *arXiv preprint arXiv:2111.11850*, 2021.

[62] F. Restuccia, S. K. Das, and J. Payton, “Incentive mechanisms for participatory sensing: Survey and research challenges,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 2, pp. 1–40, 2016.

[63] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij, “A survey of incentive techniques for mobile crowd sensing,” *IEEE Internet of Things journal*, vol. 2, no. 5, pp. 370–380, 2015.

[64] P. K. R. Maddikunta, Q.-V. Pham, D. C. Nguyen, T. Huynh-The, O. Aouedi, G. Yenduri, S. Bhattacharya, and T. R. Gadekallu, “Incentive techniques for the internet of things: a survey,” *Journal of Network and Computer Applications*, vol. 206, p. 103464, 2022.

[65] L. J. Ratliff, R. Dong, S. Sekar, and T. Fiez, “A perspective on incentive design: Challenges and opportunities,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 305–338, 2019.

[66] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, and J. Li, “Incentive mechanisms for federated learning: From economic and game theoretic perspective,” *IEEE transactions on cognitive communications and networking*, vol. 8, no. 3, pp. 1566–1593, 2022.

[67] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, “A survey of incentive mechanism design for federated learning,” *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1035–1044, 2021.

[68] X. Huang and J. Wang, “Incentive mechanisms for federated learning: From economic and game theoretic perspective,” *IEEE Transactions on Network Science and Engineering*, 2022.

[69] M. C. A. Ahmed and T. Zhang, “A systematic review of federated learning incentive mechanisms and associated security challenges,” *ACM Computing Surveys*, 2023.

[70] N. Zhao, Y. Pei, Y.-C. Liang, and D. Niyato, “Multi-agent deep reinforcement learning based incentive mechanism for multi-task federated edge learning,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 10, pp. 13 530–13 535, 2023.

[71] P. Bolton and M. Dewatripont, *Contract theory*. MIT press, 2004.

[72] Y. Zhang, M. Pan, L. Song, Z. Dawy, and Z. Han, “A survey of contract theory-based incentive mechanism design in wireless networks,” *IEEE wireless communications*, vol. 24, no. 3, pp. 80–85, 2017.

[73] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. IEEE, 2019, pp. 1–5.

[74] Y. M. Saputra, D. N. Nguyen, D. T. Hoang, T. X. Vu, E. Dutkiewicz, and S. Chatzinotas, "Federated learning meets contract theory: Economic-efficiency framework for electric vehicle networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2803–2817, 2020.

[75] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[76] L. Li, X. Yu, X. Cai, X. He, and Y. Liu, "Contract-theory-based incentive mechanism for federated learning in health crowdsensing," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4475–4489, 2022.

[77] S. Wang, H. Zhao, W. Xia, W. Wen, B. Wang, and H. Zhu, "Contract theory based incentive mechanism for clustered federated learning," in *2023 IEEE 23rd International Conference on Communication Technology (ICCT)*. IEEE, 2023, pp. 909–913.

[78] D. Yang, Y. Ji, Z. Kou, X. Zhong, and S. Zhang, "Asynchronous federated learning with incentive mechanism based on contract theory," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2024, pp. 1–6.

[79] A. Ali, I. Ilahi, A. Qayyum, I. Mohammed, A. Al-Fuqaha, and J. Qadir, "A systematic review of federated learning incentive mechanisms and associated security challenges," *Computer Science Review*, vol. 50, p. 100593, 2023.

[80] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, and D. Niyato, "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9575–9588, 2020.

[81] M. Yu and S. H. Hong, "A real-time demand-response algorithm for smart grids: A stackelberg game approach," *IEEE Transactions on smart grid*, vol. 7, no. 2, pp. 879–888, 2015.

[82] D. Yang, G. Xue, J. Zhang, A. Richa, and X. Fang, "Coping with a smart jammer in wireless networks: A stackelberg game approach," *IEEE Transactions on Wireless Communications*, vol. 12, no. 8, pp. 4038–4047, 2013.

[83] L. A. Julien, "Stackelberg games," in *Handbook of Game Theory and Industrial Organization, Volume I*. Edward Elgar Publishing, 2018, pp. 261–311.

[84] P.-y. Nie and P.-a. Zhang, "A note on stackelberg games," in *2008 Chinese Control and Decision Conference*. IEEE, 2008, pp. 1201–1203.

[85] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A stackelberg game perspective," *IEEE Networking Letters*, vol. 2, no. 1, pp. 23–27, 2019.

[86] G. Xiao, M. Xiao, G. Gao, S. Zhang, H. Zhao, and X. Zou, "Incentive mechanism design for federated learning: A two-stage stackelberg game approach," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1738–1751, 2023.

[87] W. He, H. Yao, T. Mai, F. Wang, and M. Guizani, "Three-stage stackelberg game enabled clustered federated learning in heterogeneous uav swarms," *IEEE Transactions on Vehicular Technology*, 2023.

[88] H. Zhou, T. Li, J. Li, and H. Zhou, "Federated learning based on stackelberg game in unmanned-aerial-vehicle-enabled mobile edge computing," *IEEE Transactions on Mobile Computing*, 2023.

[89] Y. Chen, H. Zhou, T. Li, J. Li, and H. Zhou, "Multifactor incentive mechanism for federated learning in iot: A stackelberg game approach," *IEEE Internet of Things Journal*, 2023.

[90] R. Zeng, C. Zeng, X. Wang, B. Li, and X. Chu, "Incentive mechanisms in federated learning and a game-theoretical approach," *IEEE Network*, vol. 36, no. 6, pp. 229–235, 2022.

[91] Texas Instruments, "Cmos power consumption and cpd calculation," Texas Instruments, Tech. Rep. SCAA035, 1997. [Online]. Available: https://www.ti.com/lit/pdf/scaa035

[92] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6360–6368, 2020.

[93] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[94] R. B. Myerson, *Game Theory: Analysis of Conflict*. Cambridge, MA: Harvard University Press, 1991.

[95] Y. Xu, M. Xiao, H. Tan, A. Liu, G. Gao, and Z. Yan, "Incentive mechanism for differentially private federated learning in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 6927–6939, 2021.