Strategic and Blockchain-based Market Decisions for Cloud Computing

Mona Taghavi

A Thesis

In

Concordia Institute for Information and Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Information Systems Engineering) at

Concordia University

Montréal, Québec, Canada

February 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mona Taghavi**

Entitled: **Strategic and Blockchain-based Market Decisions for Cloud Computing**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. Ala Al-Fuqaha

_____ Dr. Juergen Rilling

_____ Dr. Rachida Dssouli

_____ Dr. Roch Glitho

_____ Dr. Jamal Bentahar

_____ Dr. Hadi Otrok

Approved by _____
Dr. Mohammad Mannan Graduate Program Director

April 10, 2021
Date of Defence

_____
Dr. Mourad Debbabi, Gina Cody School of Engineering and Computer Science

ABSTRACT

Strategic and Blockchain-based Market Decisions for Cloud Computing

Mona Taghavi, Ph.D.

Concordia University, 2021

The cloud computing market has been in the center of attention for years where cloud providers strive to survive by either competition or cooperation. Some cloud providers choose to compete in the market that is dominated by few large providers and try to maximize their profit without sacrificing the service quality which leads to higher user ratings. Many research proposals tried to contribute to the cloud market competition. However, the majority of these proposals focus only on pricing mechanisms, neglecting thus the cloud service quality and users satisfaction. Meanwhile, cloud providers intend to form cloud federations to enhance their services quality and revenues. Nevertheless, traditional centralized cloud federations have strict challenges that might hinder the members' motivation to participate in, such as formation of stable coalitions with long-term commitments, participants' trustworthiness, shared revenue, and security of the managed data and services. For a stable and trustworthy federation, it is vital to avoid blind-trust on the claimed SLA guarantees from the members and monitor the quality of service considering the various characteristics of cloud services. This thesis aims to tackle the issues of cloud computing market from the two perspectives of competition and cooperation by: 1) modeling and solving the conflicting situation of revenue, user ratings and service quality, to improve the providers position in the market

and increase the future users' demand; 2) proposing a user-centric game theoretical framework to allow the new and smaller cloud providers to have a share in the market and increase users satisfaction through providing high quality and added-value services; 3) motivating the cloud providers to adopt a coopetition behavior through a novel, fully distributed blockchain-based federation's structure that enables them to trade their computing resources through smart contracts; 4) introducing a new role of oracle as a verifier agent to monitor the quality of service and report to the smart contract agents deployed on the blockchain while optimizing the cost of using oracles; and 5) developing a Bayesian bandit learning oracles reliability mechanism to select the oracles smartly and optimize the cost and reliability of utilized oracles. All of the contributions are validated by simulations and implementations using real-world data.

ACKNOWLEDGEMENTS

It is a genuine pleasure to express my deep sense of thanks and gratitude to my supervisors, Dr. Jamal Bentahar and Dr. Hadi Otrok. Their dedication and keen interest, timely advice, and scientific approach have helped me accomplish my Ph.D. studies.

I owe a deep sense of gratitude to my committee members and external examiner for their guide through every stage of my research. Their inspirations, timely suggestions with kindness and enthusiasm have enabled me to complete my thesis.

It is my privilege to thank my husband, Kaveh Bakhtiyari, for his scientific and academic collaboration and more importantly his constant encouragement throughout my research period.

I am extremely thankful to my parents, and brothers as well as my research lab colleagues for providing me necessary support.

This research would not have been possible without the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) that granted me the Vanier graduate research scholarship. Furthermore, I would like to thank Concordia University for all the research facilities that have been provided to me to carry out this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In this chapter, we discuss the research context and problem statements, and formulate the research questions, consequently. We further summarize the PhD research objectives to be accomplished. The chapter ends by providing the thesis organization.

1.1 Context of Research

Cloud computing is greatly perceived to be an evolutionary technology following the promising results and success stories of leading companies such as Amazon's EC2 and Google's AppEngine. A large number of services in the market inevitably incurs the competition among service providers that offer similar functionality [15]. As a result of this competition, online rating systems have attracted users' attention as an evaluation factor of providers' operational premises and their actual performance. These ratings reflect users satisfaction in today's commercial world [16] and can possibly have a large effect on the provider's revenue. The aim of this thesis is to study and investigate this effect.

The healthy competition of cloud providers in the market is threatened since

the market is dominated by only few large providers. As reported by the Synergy Research Group 2017[1], Amazon, Microsoft, Google, and IBM are gaining ground in the market at the expense of smaller cloud providers. Consequently, compatibility with private clouds and offering personalized added-value services by resellers [7] are compromised. For a hybrid of public and private cloud model to operate, a high level of compatibility between the software that runs the clouds and the business services is required that demands a higher level of customization. Thus, for the sake of the growth of the cloud computing industry, we argue that the market should be open to the new and smaller providers to create a more competitive environment. The mechanisms of such a market are yet to analysed.

The demand variation has forced cloud providers to preserve a massive amount of computing resources to avoid Service Level Agreements (SLA) violation. Resource adaptation strategies require a trade-off between the cost and performance to avoid the gap between the actual and ideal resource provisioning. If this gap is not managed properly, it can negatively impact the reputation and aggregated utility of the cloud providers in the short time, and the cloud consumers in the long run [54]. To mitigate the issue of underutilized and over provisioned computing resources, cloud providers scaled their pool of resources by forming cloud federations to maximize their profit and provide guaranteed Quality of Services (QoS) [14, 32, 47]. However, there are many partnership and trustworthiness challenges surrounding traditional cloud federations that prevent providers participation.

For a stable federation, it is vital to monitor the QoS and ensure that SLA conditions are met, since cloud providers may have an incentive to deviate. This verification is highly desirable considering the multi-tenancy characteristic of cloud services. In this context, to scale the economic benefits and optimize resource

[1] `www.srgresearch.com`

utilization, multiple VMs are initiated on the same physical server simultaneously. The performance variation depends on the network load and usage peak from other tenants. Cloud providers try to balance the workloads and achieve the required performance with less preserved capacity, yet they might not be able to supply a consistent performance. This thesis not only facilitates cloud providers collaboration within a proposed distributed and trustless federation, but also introduces novel approaches to monitor services quality.

1.2 Research Questions

Nowadays, online market and rating platforms made it easy for users to compare a wide range of services and for cloud providers to establish their own credibility. High rating comes with a price for service providers since rating represents users' benefits and not necessarily providers' profit. The problem with the existing revenue maximization strategies in the domain of cloud computing is that they do not consider the impact of the users' preferences and priorities over the price and QoS trade-off on their demands. This may result in considerable losses for providers in terms of the gained revenue and for users in terms of the quality of service. Further, it can lead to poor cloud scalability, resulting in failing providers to scale up or scale down their resources on time and to support their long-term and strategic needs. Meanwhile, the service users seek for less costly and high-quality services to optimize their own utility. The above-discussed challenges lead us to our first research question:

RQ1- *How to model the conflicting interests of the cloud service providers in terms of pricing and quality from one hand, and cloud service consumers' utility from the other hand, to maximize the providers' profit and satisfy the users' expectations to maintain their good reputation (rating)?*

The public cloud market is dominated by few large providers, which prevents a healthy competition that would benefit the end-users. It also hinders compatibility with private clouds and prevents offering personalized added-value services by resellers [7]. We argue that to make the cloud market more competitive, new providers, even small ones, should be able to enter this market and find a share. Furthermore, today's market of Cloud 1.0 is price-focused [21]. The new era of cloud computing, Cloud 2.0, has been emerged to focus on providing value to small and medium enterprises (SME) as well as large enterprise markets at higher costs as well as higher quality [21]. Despite a large number of pricing competition models [84], [25], [91], to the best of our knowledge, no one tackled the issue of the cloud providers competition from the perspective of service quality and end-user satisfaction. Therefore, our second research question is:

RQ2: *How to enable a productive cloud market industry and address the problem of cloud market share taking into consideration the need for new cloud providers in the market, and the requirements of Cloud 2.0?*

Another service quality and revenue maximization solution is cloud federation. In spite of the prominent federation advantages, cloud providers are reluctant to participate in due to some strict challenges, mainly: the stability of a federation [32], a fair revenue sharing model, the presence of unknown and untrusted participants in a federation [67], security and privacy threats regarding the managed data and services as well as the creation and management of the cloud federation itself [47]. Blockchain can provide a fully distributed architecture for a cloud federation to overcome the traditional federation models' issues. However, blockchain solutions are complex and unexplored. Thus, cloud providers need to explore different business models, participants' roles, and responsibilities, and coopetition strategies to fully embrace the technical and non-technical advantages of blockchain and smart contracts. The

above mentioned restrictions of traditional cloud federations lead us to the third research question:

RQ3: *How to design a novel fully distributed blockchain-based cloud federation architecture and platform that cloud providers can embrace and what are the best strategies to maximize their profits?*

The main restriction of blockchain is that its execution environment is self-contained and can only access information present in a transaction or in the transaction history of the blockchain [93]. A new role of verifier is required to provide some types of information about the external state. Therefore, the federation requires a trusted verifier, called *oracle*, to evaluate the service quality against SLA and ensure the compliance of the quality before the payment. Oracles can be faulty, performing malicious behaviors, or unable to perform their tasks due to lack of capacity and not being honest to report their real available resources [49]. Thus, placing a smart mechanism to select the right oracles plays a significant role in a blockchain network's success. The following research question has to be addressed:

RQ4: *How to properly monitor the quality of the provided cloud services using the most qualified oracles selected by a reliable, cost-efficient and smart mechanism?*

The above defined research questions and the research area in which they are stemmed from are illustrated in Figure 1.1.

1.3 Research Objectives and Contributions

To summarize, the main problem we aim to tackle is the theoretical and empirical foundations of an open cloud computing market. The ultimate objective is to study and analyse the challenges and opportunities of this market for the service providers in order to propose strategic and innovative solutions helping to grow the cloud

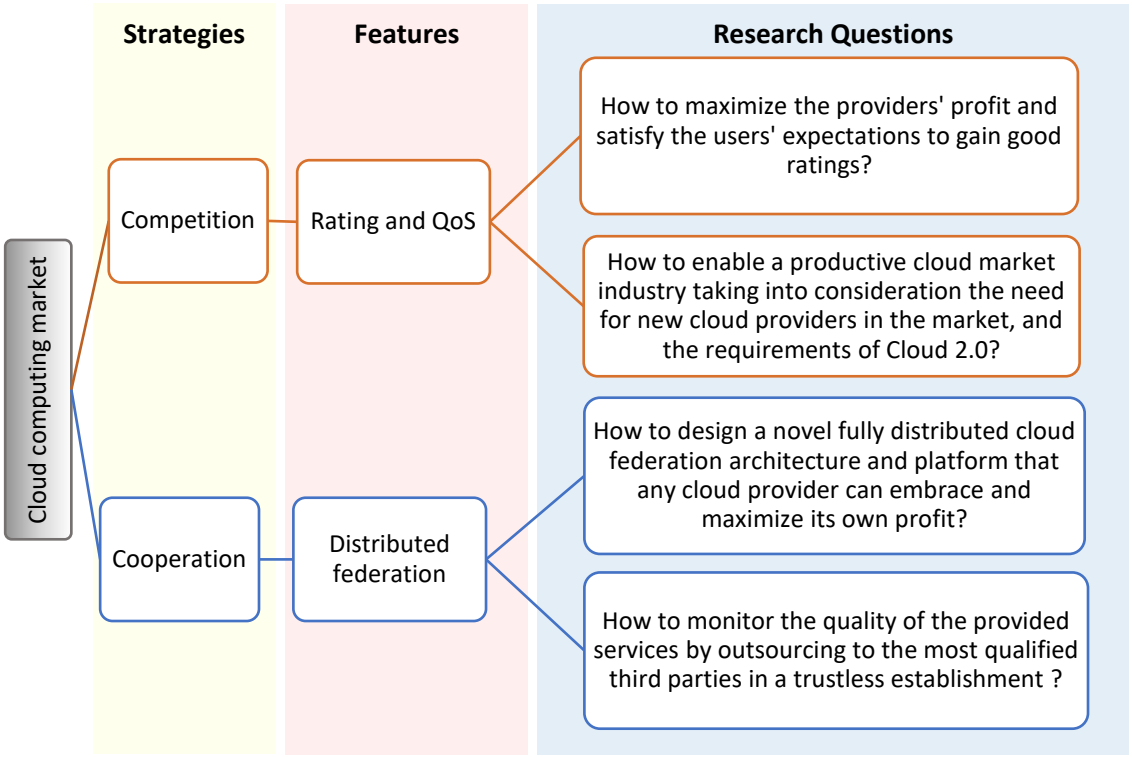


Figure 1.1: The thesis research questions

computing industry. The main contributions can be summarized as follows:

1. Assess the profitability of user ratings on cloud providers' income and identify influential parameters on users demands in a competitive online rating system. The main objective is to maximize the providers' profit through a Stackelberg game model while adjusting the services' price and capacity based on the underlining users' demand. In the meantime, the game model maintains users' satisfaction and incentivizes them to provide good ratings for the cloud providers. This contribution is published in [79] and presented in Chapter 3.

2. Design a two-stage game theoretical model considering the quality of service among cloud providers through a dynamic differential competition game. This model intends to allow new and small cloud providers to compete against the existing and large ones. It also allows to maximize users satisfaction modeled

using users' ratings by providing a continues service quality development. This contribution is published in [81] and presented in Chapter 4.

3. Advocate a fully distributed architecture with a democratic governance structure for cloud federation, using an innovative exploitation of blockchain. The architecture aims to prompt and support interoperability and coopetition among the cloud providers. Different algorithmic game theoretical models are required to help the providers make wise decisions about the utilization of the blockchain-based federation. These games focus on maximizing the profit of the formed federation's members who cooperatively compete while their service demand is dynamically changing. This contribution is published in [82] and presented in Chapter 5.

4. Introduce the new role of oracle and select the most qualified one to provide the service quality verification services and report to the blockchain to avoid the participants' misbehavior. An innovative multi-agent framework is exploited to model the roles and responsibilities of each agent involved in the cloud market including the oracles. Furthermore, a Bandit-Bayesian Learning Oracle Reliability (BLOR) mechanism is proposed to identify trustless (a term used in the context of blockchain systems meaning not requiring trust) and cost efficient oracles. The proposed mechanism also incentivizes oracles to continuously act honestly and provide a fair balance of quality and price with minimal possibility to cheat. This contribution is published in [83] and in another paper under review, and presented in Chapter 6 and Chapter 7.

As this thesis is manuscript-based, each chapter of our contributions provides a research publication as it is. An overview of the above defined objectives and the chapters in which they are addressed are presented in Figure 1.2.

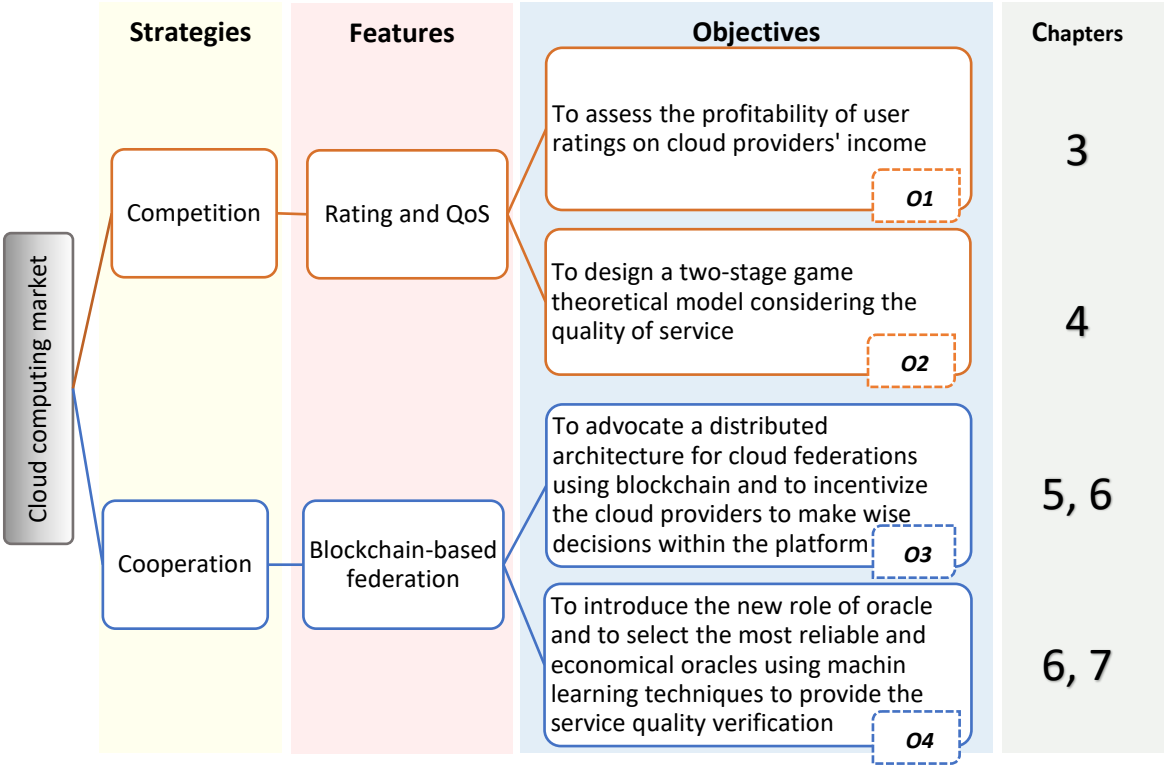


Figure 1.2: The thesis objectives and corresponding chapters

1.4 Thesis Organization

The rest of the thesis is organized as follows: the related work organized in several domains is presented in Chapter 2. In Chapter 3, a Stackelberg game model is designed to assess the profitability of user ratings for cloud providers. Chapter 4 presents a two-stage game model for maximizing the providers' profit who compete with each other over quality of service. Chapter 5 develops a blockchain-based federation using smart contracts and model the participants interactions as a differential game. Chapter 6 introduces oracles as service quality verifiers and Chapter 7 guides how to select the best oracles using reinforcement learning and a reputation model. Finally, in Chapter 8, we provide our conclusions.

Chapter 2

Research Background

In this chapter, we summarize the relevant background for this thesis from four main perspectives: 1) cloud services; 2) market share dynamics; 3) blockchain; and 4) reinforcement learning. The main topics from the literature covered in this thesis are illustrated in Figure 2.1. More details about related work are provided in the following chapters where the relevant contributions are presented.

2.1 Cloud Services

Cloud computing enables businesses to use scalable services on demand with a variety of options in pricing and quality. Following the promising results of leading companies such as Amazon's EC2, cloud computing is considered an evolutionary technology. The ease of access to cloud services over the Internet in any place and highly scalable computing resources in a shared configurable pool have made it more appealing for users specifically from small and medium companies. The advantages of cloud computing services have attracted the researchers attentions for many years and there are many research about the technical aspects of cloud computing, but

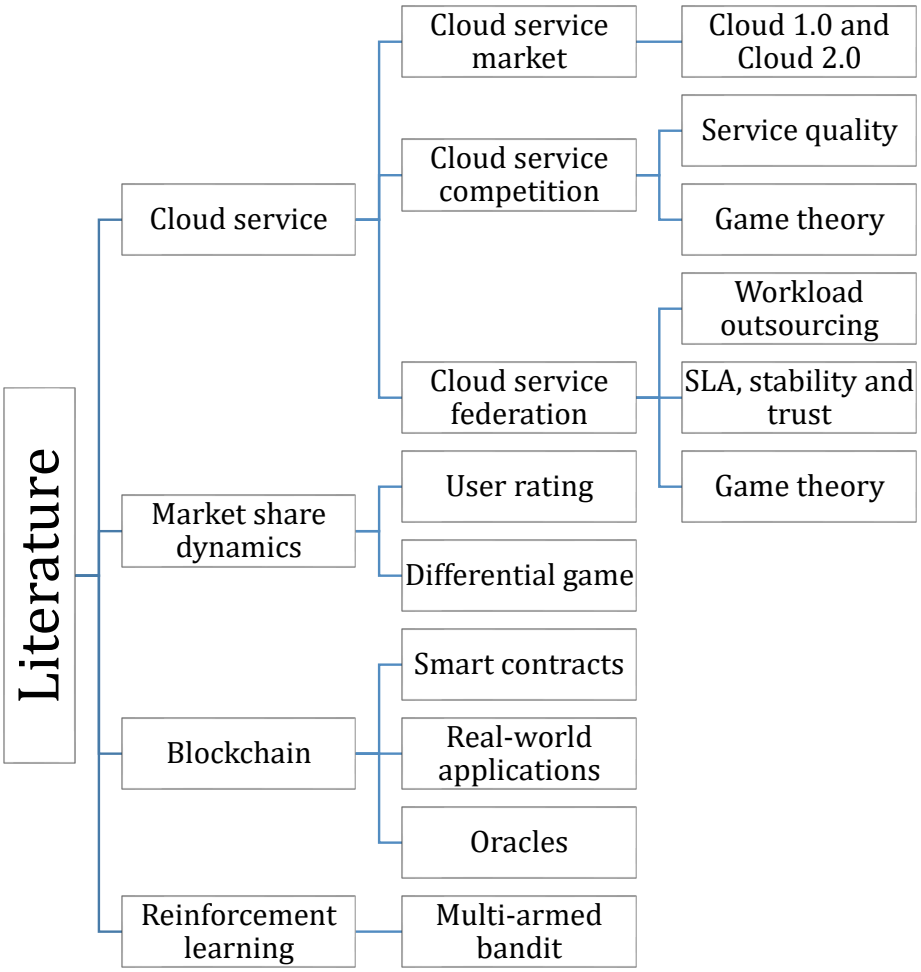


Figure 2.1: The background and literature topics covered in the thesis

very few research about its business aspects. In this thesis, we cover the research background and literature in the following areas:

Cloud Service Market: Today's cloud market, known as Cloud 1.0, is price-oriented [21]. Majority of research conducted in cloud computing market design is about pricing competitions and optimal pricing strategies to grow the revenue of cloud providers [91]. However, a price-focused service model does not suit the advanced modern business applications. The first-generation cloud providers who hide their operations details behind low pricing models made users unsure to move their critical business process

to the cloud [21].

Cloud 2.0 that represents the new era of cloud computing, pays a major attention on providing value to businesses at higher costs but superior quality [21]. Cloud 2.0 requires two transformations to happen: 1) cloud providers must provide a value that entices the businesses out of their built-in IT resources and applications; and 2) cloud customers must ask for fast, secure, and reliable cloud services from the providers to meet their end users' expectations. As an example of a cloud provider moving towards cloud 2.0, SITA[1] is an IaaS provider that offers mobility-friendly on-demand hosting and application services specifically designed for the air transport industry. SITA has connected more than 160 airports which enabled the organization to host applications accessing to airports systems, such as terminals, gates and parking.

Cloud Service Competition: Each cloud provider offers similar services at different prices and performance levels with different sets of features. For example, Amazon EC2 offers IaaS services of the same computing capabilities at different pricing for different regions. The large number of services promotes the competition among service providers offering similar functionality [15]. Survival for new and low reputed cloud providers in this competition is a big challenge.

Game theory has been explored in the cloud computing area, for instance in resource allocation and pricing mechanisms in which the interactions of players have to be taken into account [65]. Hadji et al. [31] took a user-provider interactive approach and designed a Stackelberg game to optimize the pricing of cloud services with limited resources. The cloud service providers competition is taken into account by Xu et al. [91] by optimizing a pricing policy for a better competition under the evolution

[1] `https://www.sita.aero/`

of the cloud market. Forming a Stackelberg game, the authors applied reinforcement learning (Q-learning) to find out an optimal policy for the leader provider and then for the followers.

Majority of research proposals paid a major attention to the pricing issue and somehow neglected the importance of QoS and user satisfaction. A static model for price-quality trade-off in two cases of monopoly and duopoly price competitions is proposed by Kilcioglu et al. [42]. In that model, the IaaS marketplace is referred to as commoditized from the perspective of economic competition since cloud providers use similar physical hardware, which cannot be differentiated from each other and profit margins should be omitted. The conducted experiment explained the price cutting behavior of the current market trend and also how providers are able to make a profit despite predictions that the market should be totally commoditized. Conversely, this thesis emphasizes a different approach aligned with the vision of Cloud 2.0. Young and small competitors cannot survive in the market with commoditization because of their lower number of users and higher expenses. We argue that smaller providers need to differentiate themselves from the established large providers in the market by providing added-value services to their customers. Fan et al. [24] performed an inspiring study on cloud service quality by considering market competition among a Software as a Service (SaaS) provider and a traditional software provider as a differential game. This research analyzes a short and long-term competition for price and dynamic quality between the two firms.

Cloud Service Federation: Cloud service federations have been formed in response to users demands variation and shortage of resources. When it comes to federation formation, coalitional games are mainly entertained in the literature where the providers resource capacities and their revenues are shared [63]. Federation-formation

variables, including revenue sharing mechanisms, capacity and cost disparity, and the presence of a big competitor was invesstigated by Coronado et al. [17]. They defined revenue sharing mechanisms as the most important factor. Among these mechanisms, shapely value and outsourcing models (a provider outsources some of its business and gets a percentage of the revenue) had the least and best performance, respectively. The authors found that cloud providers tend to stay in outsourcing collaboration when the user demand is high. Their findings confirm the superiority of outsourcing in terms of maximizing the profit of cloud providers, which is what we are proposing in this thesis using a blockchain platform considering interoperability, trust among cloud providers and service quality, which are not investigated in their study. Chen et al. [14] proposed another cloud outsourcing model using the coalition formation game among private clouds and found to be promising to improve the cloud's service quality. Zhao et al. [103] investigated the cost-efficiency of data centers and the cloud providers' revenue considering the impact of the two factors of energy consumption and SLA violations. They developed online VM placement algorithms as an optimization problem of maximizing revenue from VM migration. However, no initiatives are proposed to monitor the SLA violations. The dynamic and timed decision making strategies are also not considered.

2.2 Market Share Dynamics

The market share of a company is the percent of total sales in an industry generated by the firm over the period in which the firm operates. This metric shows how large a company is in comparison to its industry and its competitors. In this thesis, we study the dynamics of service price, quality, and market share of the providers in an oligopoly market. Such a knowledge enables the cloud providers to learn

their customers behaviors as well as their competitors to plan their market position strategies to gain more share in the market. We consider a cloud computing market in which consumers can learn about the services in two ways: use their own experience, or check their users' satisfaction reflected through users ratings. The privilege of users rating and its impact on providers income play a key role in providers incentives to determine not only their short-term, but also their long-term strategies. Therefore, we need to define different types of dynamics in our model.

Many proposals in the literature consider static market share [33]. Some of the non-static studies exercised differential games to formulate the market share dynamics. For instance, Breton et al. [10] studied dynamic equilibrium advertising strategies in a duopoly market using differential Stackelberg game. In another attempt, Gutierrez et al. [30] analyzed the dynamic strategic interactions between a manufacturer with the assumption that the retail demand is influenced by word-of-mouth from past adopters. The obtained equilibrium dynamic pricing showed that in some cases, far-sighted retailer is more profitable. Even though these studies help businesses optimize their sale, they disregard the customer satisfaction effect on market share.

Currently, major companies such as Amazon, Netflix, Launch, Google, YouTube and Facebook are heavily using and relying on user ratings to sell their services, leading to a significant increase of revenue [80]. Today's on-line market enables service providers to establish their own credibility. In spite of the acknowledged importance of users' ratings in marketing strategies [80], only few proposals investigated its effect on business owners income [16]. For example, Duan et al. [20] studied video sales and movie recommender systems and found that users' ratings increase the users' awareness by word-of-mouth and increases the providers sales directly. Completing their study, our research proves that cloud service quality significantly affects the overall users' ratings, enhances the providers reputation and increases their profit.

2.3 Blockchain

Blockchain was initially conceived as an anonymous and trustless peer-to-peer system used for financial transactions and has evolved over the years to include a wide variety of other applications such as smart contracts [96]. In fact, blockchain offers a distributed ledger to track and sustain a record of transactions across a decentralized network. The distributed ledger contains all verified and validated transactions in a verifiable, secure, transparent, and lasting method along with a timestamp. Each stakeholder keeps a copy of the ledger, so it can't fail at one point. When changes are recorded, such as adding a block, they are simultaneously updated in all copies across the network, and records are permanently registered in all ledgers. Changes are stored as blocks, which build up a chain, where each block is linked to the previous by storing its hash.

Figure 2.2 presents the chained architecture of blockchain and the smart contract and oracles interaction over the blockchain. Excluding the first block (called Genesis), each block has its unique ID and includes the hash of the previous block. For example, block 186 stores the hash code of block 185 besides its own hash. This method results in the formation of a chronological chain. Additionally, the data is more secure using the hash algorithm. Usually, a blockchain consists of a set of transactions $[0 - N]$ that have been time-stamped and are validated by stakeholders within the network. Once the block gains consensus, it becomes a part of the blockchain and is no longer allowed to be modified. Therefore, trust and transparency are significantly improved in communications between its participants. Smart contracts are executable codes residing on blockchain networks that contain agreements among its users. Smart contracts use oracles to fetch data from the outside world. As Figure 2.2 shows, when a user's query requires data from outside of the blockchain network, the smart contract asks the oracle to retrieve the data within a new self-created contract. Then,

the oracle collects the data from an off-chain source and transfers it on-chain with a signed message and makes the data available by putting it in the smart contract's storage.

Blockchain-based platforms are categorized as a permissionless or permissioned blockchain [55]. Anyone can participate anonymously in a permissionless blockchain network, also called public blockchain. Within a public blockchain, trust is limited, so that miners are introduced to validate the registered transactions. On the contrary, permissioned blockchain, called private blockchain, contains a group of identified users who are trusted. To join this type of network, new users need permission from the majority of the group or a delegated user. Miners collect and validate broadcasted transactions and create new blocks. They compete against each other to solve a mathematical puzzle, widely known as a proof-of-work problem. The first who solves the puzzle, creates a new block to the chain and earns a specific amount of reward, such as a small number of Bitcoins.

Smart Contracts: A smart contract defines the conditions and obligations among stakeholders and lives on the blockchain with a unique address [104]. The information of a smart contract is recorded as an executable computer code that enables it to self-execute when all of the predetermined conditions are satisfied within a blockchain network. Therefore, stakeholders who make their agreements upon a smart contract build stronger trust relation and are less likely to face any fraud. Moreover, smart contracts provide users with the freedom to build autonomous applications that function independently of the system entities.

The most popular blockchain with smart contract functionality is called Ethereum and its cryptocurrency is Ether. In the current version, Ethereum functions through gas which is an Ether-based purchase of the consumed resources.

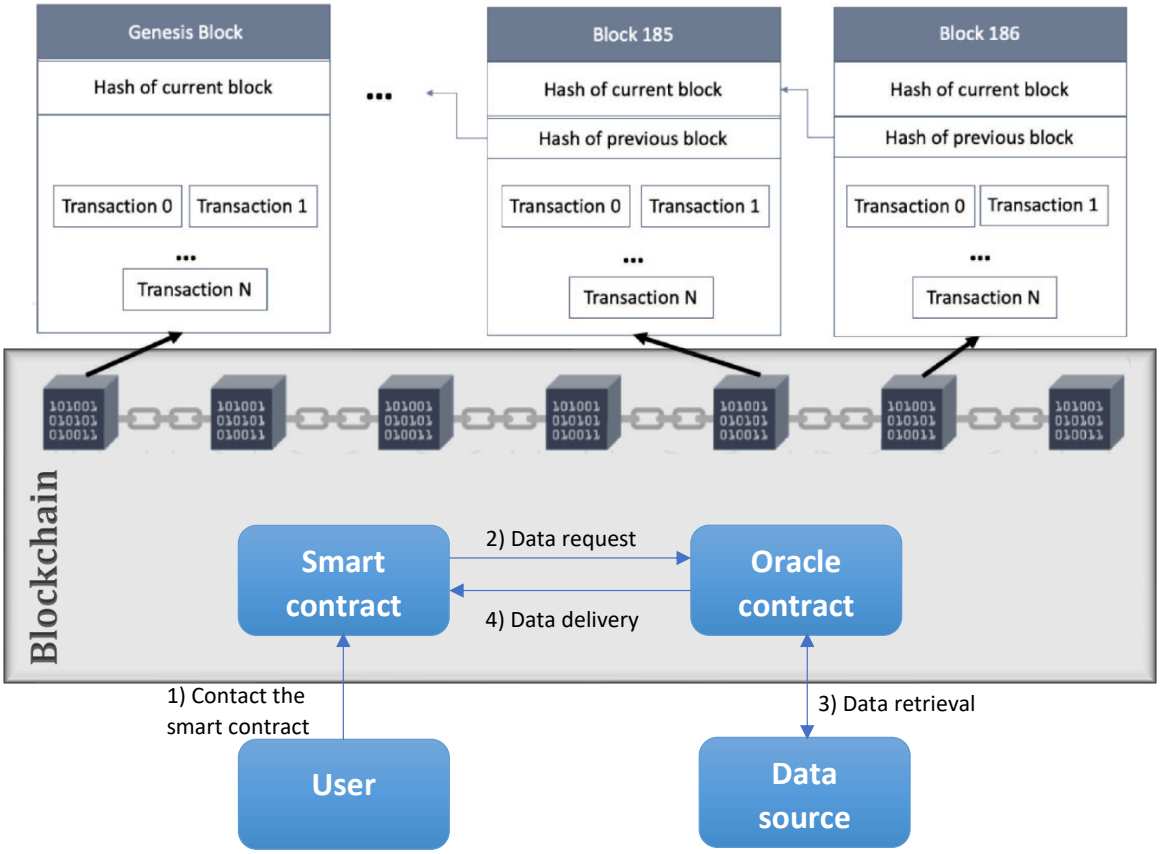


Figure 2.2: Blockchain and smart contract architecture

This will help Ethereum prevent DoS attacks, infinite loops within contracts, and network resource expenditure. Every function, such as sending and retrieving data, executing computation, and storing data, has a gas cost. Smart contracts have two types: deterministic and nondeterministic [57]. The deterministic smart contract is implemented on a blockchain with complete isolation from external environments, and participants are responsible for maintaining the contract states and decisions. On the other hand, nondeterministic smart contracts require external information to make decisions, which makes them dependent on actors outside of the blockchain network. For instance, an external actor could be a weather provider or a sensor data provider, who are referred to as oracles in blockchains.

Oracles: Blockchain oracles provide a link between off-chain and on-chain data. Apparent absence of oracles would limit the smart contracts functionality as they would only have access to data from within their networks [27]. Whether an oracle depends on human involvement or is totally automated determines the process by which it obtains its data. Automated oracles operate solely through software and hardware to fetch the data and oracle itself is not the original source of the data. Automated oracles only provide deterministic inquiry results. However, autonomous oracles or oracles involving human intervention are able to respond to arbitrary inquiries hard to be deducted by machine. They cannot be distinctly separated from the data source. Until writing this thesis, no paper was found addressing the reliability of such oracles.

Oracle systems can be centralized or decentralized. Oraclize [2] is a centralized oracle service based on Amazon Web Service with main attention of proving that the obtained data is untampered. Town Crier [99] is also a centralized authenticated data feed that operates as a trusted bridge between existing HTTPS-enabled websites and Ethereum using trusted hardware and software. However, similar to any other centralized solution, their validity relies on a central authority and the correctness of the original source or the performed task is questionable. Chainlink [23] is a decentralized oracle network on the Ethereum platform aiming to provide tamper-proof data using designated APIs. Chainlink operates through incentives and aggregation models, however, it has cost and scalability issues. ASTRAEA [6], is an interesting decentralized oracle working based on a voting game to decide about the truthfulness of propositions. The authors analyzed the game-theoretical incentive structure to prove the existence of Nash equilibrium under the assumption that all rational players behave honestly.

[2] `https://provable.xyz/`

Real-World Applications: It has been known for some time but there is little study on the potential of blockchain in real-world applications despite its vast potential for business sharing data and collaborating in a secure and customizable way [53]. Blockchain applications mainly focus on finance [85], energy [60] and IoT applications [102]. In cloud computing and service industry, to the best of our knowledge, there has been only one academic initiative that proposed a cloud marketplace based on the blockchain technology, called Desmaa [43]. Desmaa is a conceptual framework for trustless intermediation in service marketplaces that models the interactions between a service provider and a service consumer. Yet, the outsourcing model with collaboration and competition among cloud providers, the providers' profit and their best strategies, and evaluation and validation against real-world's scenarios are not elaborated.

2.4 Reinforcement Learning

Reinforcement learning was developed as an approach to mimic the behavior of a biological agent [77]. As Figure 2.3 illustrates, reinforcement learning uses three elements: observation s_t, action a_t, and reward r_t at time step t to solve a problem. The idea is that an intelligent agent learns how to optimize its actions based on how it is rewarded for its actions in the current and resulting states. It can then optimize its actions to accumulate more rewards towards its desired objective. This approach can be applied to any sequential decision-making problem based on the learned policy. The environment which an agent is interacting with is modeled as a Markov Decision Process that could be completely or partially observable.

Reinforcement learning has two main elements :1) agent and 2) environment,

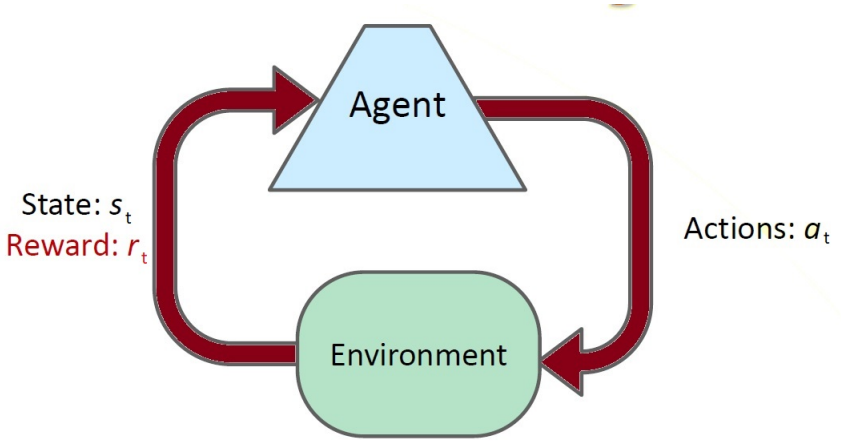


Figure 2.3: Blockchain and smart contract architecture

and three sub-elements named reward, policy and value function. Maximizing the reward is the goal of the reinforcement learning problem. Policy is a mapping of states and actions. It defines what actions should be taken in each state by the agent within the environment. Value function defines the optimal value in a long run. Unlike reward signal, it focuses on long term return of selected actions.

Reinforcement learning methods are classified as model-based and model-free. Model-based learning involves modeling of the transitions and immediate outcomes in the environment and choosing the optimal policy based on the model. Model-free learning uses trial and error experiences to build the optimal policy. As the state space and action space grow, model-based algorithms become impractical. In contrast, model-free learning does not require space to store all the combination of states and actions tailed in a model. The most widely used algorithms of model-free learning method are temporal difference, Q-learning and multi-armed bandit.

Q-Learning: The goal of Q-learning is to learn a policy that guides an agent what action to take under what circumstances. For any finite Markov decision process, Q-learning finds a policy to maximize the expected value of the total reward over

any and all successive steps, starting from the current state. Q-learning can identify an optimal action-selection policy for any given finite Markov decision process, given infinite exploration time and a partly-random policy. "Q" names the function that returns the reward used to provide the reinforcement and stands for the "quality" of an action taken in a given state.

Temporal Difference (TD): TD is a learning method that depends on the future value without any prior knowledge. The name of temporal difference is derived from the concept of change and difference in various stages to update its prediction of the future value, which leads to a learning process. In fact, TD integrates some of the features of both Monte Carlo and Dynamic Programming (DP) methods and has the advantage of being implemented in an on-line fully incremental fashion over these methods.

Multi-Armed Bandit: It is a problem in which there are fixed number of choices and the objective is to maximize the gain by selecting the best choice. In this context, each choice can be partially observed and they may be better explored and identified as the time goes on by examining them on multiple occasions. Multi-armed bandit has its origins in slot machines, in which players must determine which machine to play, how many times and in what order to play it to maximize their prizes. In this context, each machine gives a stochastic reward from a probability distribution. Multi-armed bandit problem is exploited in many fields such as medical [86], recommender systems [48], and crowdsourcing [38]. However, to the best of our knowledge, its application in blockchain together with its specific challenges have not been explored in any research yet. Multi-armed bandit is highly accurate and requires little complexity and processing resources when compared to other reinforcement learning methods. This made it an ideal solution to be utilized on blockchains and smart contracts in which computation and storage are very precious.

Chapter 3

On the Effects of User Ratings on the Profitability of Cloud Services

In today's cloud market, providers are taking advantage of consumer reviews and ratings as a new marketing tool to establish their credibility. However, to achieve higher ratings, they need to enhance their service quality which comes with an additional cost. In this chapter that presents our published manuscript, we model this conflicting situation as a Stackelberg game between a typical service provider and multiple service users in a cloud environment. The strategy of the service provider is to adjust the price and IT capacity by predicting the users' ratings as well as their demands variation in response to his given price, quality and rating. The game is solved through a backward induction procedure using Lagrange function and Kuhn-Tucker conditions. To evaluate the proposed model, we performed experiments on three real world service providers who have low, medium and high average of

users' ratings, obtained from the Trust Feedback Dataset in the Cloud Armor project. The results show that improvement in ratings is mostly profitable for highly rated providers. The surprising point is that providers having low ratings do not get much benefit from increasing their average ratings, meanwhile, they can perform well when they lower the service price. This chapter is published in [79].

3.1 Introduction

Cloud computing has emerged as a significant promising computing paradigm by facilitating customers access to computing services without owning any computing resources. The large number of services inevitably incurs the competition among service providers that offer similar functionality [15]. Survival for new and less famous cloud providers in this competition is more challenging, unless they provide high quality services and gain good reputation. Today's on-line market made it easy for providers to establish their own credibility.

On-line rating systems have attracted users' attentions as an evaluation factor of providers' operational premises and their actual performance. Rating platforms enable users to share their experience and interests with other users in a timely fashion. Reviews and ratings, known as digitized word-of-mouth, play an important role in the future customers decision making. Theses ratings reflect users satisfaction in today's commercial world and can affect the providers revenue largely [16].

High rating comes with a price for service providers, since rating represents users' benefits and not necessarily providers' profit. The main issue arises due to the fact that each participating party in the cloud has its own interest. Users want to purchase elastic and high-quality services with minimum price. However, from the provider's perspective, higher quality means more cost and minimum price means low

profit. Moreover, the service price has a large effect on users willingness to order, and quality influences users' ratings that represent their satisfaction as a reference for future users. Planning a suitable pricing strategy in early stages of the service development life cycle is highly significant since pricing may give special requirements to the architectural design, such as scalability and customizability [46]. The problem with the existing revenue maximization strategies in the domain of cloud computing is that they do not consider the impact of the users' preferences and priorities over the price and QoS trade-off on their demands. This may result in considerable losses for providers in terms of the gained revenue and for users in terms of the quality of service. Further, it can lead to poor cloud scalability failing providers to scale up or scale down their resources on time, and to support their long-term and strategic needs. Failing to meet the expected users' demands for cloud services can result in deficiency or large up-front investments in infrastructure. To address these issues, we model the conflicting interests and selfish actions of the participants as a Stackelberg game. The strategy of the service provider as a leader is to maximize his profit and, at the same time, satisfy the users' needs to maintain his good reputation. Meanwhile, the service users as followers seek for less costly and high quality services to optimize their own utility.

Contributions: The novelty of this study lies in the theoretical and empirical research conducted to study the impact of on-line customers' ratings and demand variations on the revenue of infrastructure cloud service providers. The main contributions can be summarized as follows:

- Assessing the profitability of user ratings on cloud providers' income in a competitive on-line rating system. To the best of our knowledge, our work is the first that presents a comprehensive study on the users' ratings on the providers' profit in a cloud environment.

- Enabling providers to identify influential parameters on users demands and capturing the variations of users' demands in response to the changes of each parameter to enable scalability of cloud services and avoid under and over resources provisioning.
- Maximizing the providers' profit through a Stackelberg game model while adjusting the services' price and capacity based on the underlining users' demand.
- Maintaining users' satisfaction and incentivizing them to provide good ratings for the providers.

For empirical evaluation, the model is implemented using a real world dataset obtained from the Cloud Armor project[1], on three service providers with low, medium and high rating.

3.2 Related Work

Game theory is widely applied where the interactions of players have to be taken into account. This cannot be designed with the classical optimization theory, since the players' actions affect the other players. Game theory has been successfully applied to address resource allocation and Quality of Service (QoS) issues [66]. In cloud computing it is mainly utilized to deal with resource allocation and pricing issues [65]. As an example, a two stage provisioning Stackelberg game is offered by Di Valerio et al. [18] for Software as a Service (SaaS) providers who use cloud facilities provided by an Infrastructure as a Service (IaaS) provider. First, the SaaS providers determine the number of required instances, then the SaaS providers compete by bidding for the spot instances.

[1] http://cs.adelaide.edu.au/~cloudarmor/ds.html

The perspective of the user and provider is considered by Al Daoud et al. [1], who propose a policy to maximize the cloud provider's revenue and users utilities. The authors focus on the pricing problem and proved the existence of a Nash equilibrium. A very similar approach is taken by Hadji et al. [31]. A Stackelberg game is designed to consider constrained pricing with limited resources offered by an IaaS provider and the optimal user demands. However, price is the only utility factor considered for both the user and provider in existing research; and the importance of QoS or ratings as well as the trade-off relation between price and QoS are yet to be investigated. It is worth mentioning that none of the above discussed research has utilized real world datasets for demonstration of their game applicability in real life.

A recent survey conducted by BrightLocal in November 2016, acknowledged that 84% of people trust on-line ratings and reviews as much as personal recommendations, and 58% of consumers say that the star rating of a business is the most important decisive factor[2]. Yet, there have been few works exploring the user ratings effect on business owners profit [16], that can be found mainly in marketing and economic literatures. For instance, empirical studies showed that improving book review ratings on Amazon.com and BarnesandNoble.com tends to increase their sales [16]. In the cloud service literature, Wang et al. [89] proposed a reputation measurement approach based on feedback ratings to obtain the trust vector of each cloud service. Their model generates a reputation score for cloud services and is limited to the users who already used the service in the past, but does not support future users. To the best of our knowledge, this work is the first that models cloud services profitability while considering future users ratings, where unlike classic economic models, the main challenge is how to consider the elasticity feature of cloud services along with QoS factors.

[2]www.brightlocal.com/learn/local-consumer-review-survey/

3.3 Cloud Service Provider-User Stackelberg Game

We model the cloud service market interactions between a service provider and the service users as a Stackelberg game, where the service provider is the leader and the service users are the followers. The users observe the price and ratings to adjust their demand accordingly. In quest of the users demands, the service provider makes decision on his pricing strategy and optimal capacity. In the provider objective model, the provider tries to comply with Service Level Agreement (SLA) to obtain and maintain his good rating, otherwise poor quality affects users rating and future users demand. We assumed there is no limitation for provider capacity, so he can increase his capacity as the demand grows. The proposed model considers different parameters, which are provided in Table 3.1.

The cloud service delivery requires provisioning an estimated amount of the required cloud resources to satisfy customers' demands. A precise estimation will benefit cloud providers with a balanced capacity and reduced cost. This challenging task of estimation depends on several factors including the number of consumers, variation of their demand, and their expected QoS [59]. Elasticity capabilities of cloud resources enable providers to scale their capacities and to configure provisioned resources to take into account the user demand behavior and specified QoS requirements for each user. In order to capture demand elasticities and variations specific for each user, which are fundamental aspects in cloud environments, we define the user demand using the Cobb-Douglas function that models well these elasticity aspects [29].

Table 3.1: Notations used in service provider-user Stackelberg game

Decision variables	
x_{ik}	Demand size of user i for service k
ϕ	IT capacity/process rate of the service provider
P_k	Price per unit of service k
Input parameters	
$i = 1, 2, ...n \in N$	Index of n users in the set N
B_i	User i Budget
R_{ik}	Rating utility of service k from user i
r_{ik}	Service rating of user i for service k
$\bar{r}_k$	Average of n users' ratings for service k
$\bar{r}_i$	Average ratings of all the services given by user i
$\hat{r}_{ik}$	Predicted rating of user i for service k
α_i	Price elasticity for user i
β_i	Rating elasticity for user i
γ_i	Amount of service elasticity for user i
l	User's arrival rate
k, j	Services (offered by two different providers)
μ	Constant scale of user demand
ϕ	IT capacity/process rate of service provider
Q_k	Quality of service k stated in SLA
C_{0k}/C_k	Fixed cost/marginal cost of service k
$\lambda, \lambda_1, \lambda_2$	Lagrange multipliers

The Cobb-Douglas demand function is continuous, convex or concave, and has constant elasticities in relation with each input parameter. In real world situations, a

user demand depends on service price and perceived quality. The user will have the opportunity to check the provider rating that represents the actual user satisfaction level of the service quality. Therefore, in addition to the amount of service and the service price, the user rating is considered influential in our user demand function. We define the user demand function as follows:

$$D_i(x_{ik}, P_k, r_{ik}) = \mu \; x_{ik}^{\gamma_i} \; P_k^{-\alpha_i} \; r_{ik}^{\beta_i} \tag{3.1}$$

where $\alpha_i,\ \beta_i$ and $\gamma_i,\ i = 1, 2, ..., n$ are elasticities of the service price P_k, rating r_{ik} and size x_{ik} respectively. Different market users, having different requirements and satisfaction levels, do not react evenly to the same price or rating. It is the combination of these factors that produces different values of α_i, β_i and γ_i. These values are independent of the specific values of P_k, r_{ik} and x_{ik}, which is an inherent property of the Cobb-Douglas function. User demand has a negative relation with service price, and positive relation with service rating. The user (i.e., a typical follower) aims to maximize his payoff:

$$\begin{aligned} \text{maximize} \quad & UP(x_{ik}) = D_i(x_{ik}, P_k, r_{ik}) - P_k \\ \text{subject to} \quad & P_k x_{ik} \leq B_i \\ & x_{ik} \geq 0, \forall i \in N \end{aligned} \tag{3.2}$$

The user's objective is to maximize the demand size x_{ik} within his budget B_i while minimizing the cost P_k. Users' ratings that reflect their satisfaction level enhance their total utility encoded in the demand function. Service provider predicts the new user rating based on the given previous ratings that is influenced by the actual service quality. The user can only decide on the size of the demand, and price should be obtained through the provider's utility function.

As the cloud service provider needs to maintain his reputation through the user ratings, he is responsible to process users requests on time. Thus, it is important to consider service processing rate that represents IT capacity of the service provider, denoted as ϕ. A large processing rate requires a higher IT capacity, meaning a higher cost for ϕ that includes fixed cost of C_0 and marginal cost of C_k. Thus, the total cost for capacity ϕ is $C_{0k} + C_k\phi$.

Following previous literature in cloud computing [24], we model arrival of customers as a Poisson process with mean arrival rate l. The average delay for a customer in an M/M/1 queue can be defined as $\frac{1}{\phi - l}$. The provider is willing to optimize his profit by maximizing the price and ratings given by the users, and minimizing the costs. Thus, the provider (i.e., the leader) optimization problem is:

$$\begin{aligned} \text{maximize} \quad & PP(P_k, \phi) = \sum_{i=1}^{n} (P_k - \ \phi \ C_k) \ D(x_{ik}^*, P_k, r_{ik}) \\ & + \ \sum_{i=1}^{n} R_{ik} - C_{0k} \\ \text{subject to} \quad & \frac{1}{\phi \ - \ l} \leq Q_k \\ & \phi > 0, P_k > 0, \forall i \in N \end{aligned} \tag{3.3}$$

x_{ik}^* is the outcome of the optimization problem Eq.3.2, which corresponds to the best user's response in terms of demand size to the offered service price and quality. The provider can only maintain his high records of ratings, if he offers a service quality not less than what is stated in SLA. Thus, based on the defined constraint, the average delay should not be more than Q_k stated in SLA.

The user ratings do not always enhance the utility of the provider. When the provider receives a low rating, it may have a negative effect on his payoff. To reflect that, we assign to R_{ik} a negative sign when the user rating is less than the average

user ratings as follows:

$$R_{ik}: = \begin{cases} +R_{ik} & \text{if } \hat{r}_{ik} \geq \bar{r}_k \\ -R_{ik} & \text{if } \hat{r}_{ik} < \bar{r}_k \end{cases} \tag{3.4}$$

$\hat{r}_{ik}$ is the predicted rating of user i which will be calculated in the next section.

3.4 User Rating Prediction

Each service has a history of user rating values that can be used for future user ratings for other services. In this paper, we predict the rating value of service k using a set of similar services to service k that have been rated by the users. The similar service neighbors are identified using the Pearson Correlation Coefficient (PCC) measure. PCC is a common method of similarity computation in recommender systems that measures the extent to which two variables linearly relate with each other. Therefore, the similarity among two services k and j with the same functionality consumed by user i is computed as follows:

$$Sim(k,j) = \frac{\sum_{i \in N}(r_{ik} - \bar{r}_k)(r_{ij} - \bar{r}_j)}{\sqrt{\sum_{i \in N}(r_{ik} - \bar{r}_k)^2}\sqrt{\sum_{i \in N}(r_{ij} - \bar{r}_j)^2}} \tag{3.5}$$

where $\bar{r}_k$ and $\bar{r}_j$ represent the average rating values of service k and j consumed by n users. After calculating the similarity values, it is important to select neighbors that are really similar to the service. Therefore, the similar neighbor set S for service k is defined as follows:

$$S(k) = \{j | j \in T_K, Sim(j,k) > 0, j \neq k\} \tag{3.6}$$

T_K represents a set of the Top-K similar services to service k. The identified similar service set is utilized for rating predictions. Based on the user experience of the similar service set, the missing rating value of service k for user i would be:

$$\hat{r}_{ik} = \bar{r}_i + \frac{\sum_{j \in S(k)} Sim(k,j)(r_{ij} - \bar{r}_j)}{\sum_{j \in S(k)} Sim(k,j)} \tag{3.7}$$

Predicted rating of service k from user i will be placed as an input for the defined function of R_{ik} in Eq.3.4 to compute the final utility for the service provider. For convenience, we use r_{ik} to designate $\hat{r}_{ik}$ or r_{ik} in the rest of the paper.

3.5 Stackelberg Game Equilibrium

We solve the equilibrium point of the above defined Stackelberg game by a backward induction procedure. Therefore, the followers' (users) problem has to be solved first to obtain the response function of these users. The leader's (provider) decision problem is then computed considering all possible reactions of his followers in order to maximize his net profit. For every possible provider's action, every user's optimal reaction shall be determined by considering the provider's decisions as its input parameters. At last, the provider identifies his optimal decision that leads to his optimal payoff, by assuming that the users are rational and make the optimal response to his decisions. The best response functions are discussed in the following sections.

3.5.1 User Best Response

The user has to adjust the size of his demand according to his budget for a given price. In our model, increasing the budget is not allowed for the user. By definition, the Cobb-Douglas function $D_i(x_{ik}, P_k, r_{ik})$ is an increasing and concave function of

r_{ik}, so we have a positive first derivative and a negative second derivative,

$$\frac{\partial D_i(x_{ik}, P_k, r_{ik})}{\partial r_{ik}} = \beta_i \mu P_k^{-\alpha_i} r_{ik}^{\beta_i - 1} x_{ik}^{\gamma_i} > 0 \tag{3.8}$$

$$\frac{\partial^2 D_i(x_{ik}, P_k, r_{ik})}{\partial r_{ik}^2} = \beta_i(\beta_i - 1) \mu P_k^{-\alpha_i} r_{ik}^{\beta_i - 2} x_{ik}^{\gamma_i} < 0 \tag{3.9}$$

Considering the above equations, we have $0 < \beta_i < 1$. We can get the same range for γ_i, $0 < \gamma_i < 1$, since the function is increasing and concave in x_{ik}, and $\alpha_i > 0$.

As the objective function in Eq.3.2 is continuous and concave in x_{ik}, we obtain the solution using Lagrange multipliers, λ_1 and λ_2, with Kuhn-Tucker conditions. So, we will have a new objective function:

$$L_{up} = D_i(x_{ik}, P_k, r_{ik}) - P_k - \lambda_1(x_{ik} P_k - B_i) + \lambda_2 x_{ik} \tag{3.10}$$

with the following conditions:

$$\lambda_1(x_{ik} P_k - B_i) = 0 \tag{3.11}$$

$$\lambda_2 x_{ik} = 0 \tag{3.12}$$

$$\lambda_1, \lambda_2, x_{ik} \geq 0$$

The only coupling point between users is x_{ik}, so we take the derivative with respect to x_{ik}.

$$\frac{\partial L_{UP}\ (x_{ik})}{\partial x_{ik}} = \frac{\partial D_i\ (x_{ik})}{\partial x_{ik}} - \lambda 1 P_k + \lambda_2 = 0 \tag{3.13}$$

We have two cases: 1) $x_{ik} = 0$: regardless of the value of λ_1, λ_2, this means that the user is not demanding any services, so his utility will be zero. 2) $x_{ik} > 0$: from slackness complementary condition in Eq.3.12 we can conclude that $\lambda_2 = 0$; so we have:

$$\gamma_i \mu x_{ik}^{\gamma_i - 1} P_k^{-\alpha_i} r_{ik}^{\beta_i} - \lambda_1 P_k = 0 \tag{3.14}$$

$$x_{ik} = \left(\frac{\lambda_1 P_k^{\alpha_i + 1}}{r_{ik}^{\beta_i} \gamma_i \mu}\right)^{\frac{1}{\gamma_i - 1}} \tag{3.15}$$

By substituting x_{ik} from Eq.3.15 in Eq.3.11 we obtain λ_1:

$$\lambda_1 \left[\left(\frac{\lambda_1 P_k^{\alpha_i + 1}}{r_{ik}^{\beta_i} \gamma_i \mu}\right)^{\frac{1}{\gamma_i - 1}} P_k - B_i\right] = 0 \tag{3.16}$$

$$\lambda_1^{\frac{1}{\gamma_i - 1}} = \frac{B_i r_{ik}^{\beta_i} \gamma_i \mu}{P_k^{\frac{\alpha_i + 1}{\gamma_i - 1} + 1}} \tag{3.17}$$

The final response x_{ik} from user i is attained by replacing Eq.3.17 in Eq.3.15.

$$x_{ik}^* = \frac{B_i (r_{ik}^{\beta_i} \gamma_i \mu)^{\frac{\gamma_i - 2}{\gamma_i - 1}}}{P_k} \tag{3.18}$$

The above obtained x_{ik}^* is optimal where Eq.3.11 slacks and $\lambda_1 > 0$. However, we claim that it is reasonable to consider slackness rather than binding, since having $\lambda_1 = 0$ is an extreme case where the user cares only about the price and does not consider the previous ratings or quality.

3.5.2 Cloud Service Provider Best Response

In the case of having a non zero demand for the service provider and close values of price and cost, the provider can only survive when he receives a high rating that can

cover his sacrificed price loss. But, if his rating is low and he cannot set a high price, he will eventually suffer from a loss and leave the market. Using Lagrange multiplier λ, we model the objective optimization in Eq.3.3 as follows:

$$L_{PP}(P_k, \phi, \lambda) = PP(P_k, \phi) - \lambda(\frac{1}{\phi - l} - Q_k) \tag{3.19}$$

The Kuhn-Tucker condition for our model is:

$$\frac{\partial PP(P_k, \phi)}{\partial \phi} - \lambda \frac{\partial(\frac{1}{\phi - l} - Q_k)}{\partial \phi} = 0 \tag{3.20}$$

$$\frac{\partial PP(P_k, \phi)}{\partial P_k} = 0 \tag{3.21}$$

$$\lambda(\frac{1}{\phi - l} - Q_k) = 0 \tag{3.22}$$

where $\lambda \geq 0, P_k, \phi > 0$. To find the optimal capacity ϕ, we first assume that Eq.3.22 binds and $\lambda > 0$. Referring to Eq.3.20 we have:

$$C_k D_i(x^*_{ik}, P_k, r_{ik}) - \lambda(\frac{-1}{(\phi - l)^2} = 0$$

$$\lambda = -C_k D_i(x^*_{ik}, P_k, r_{ik})(\phi - l)^2 \tag{3.23}$$

Knowing that $C_k > 0$ and $D_i(x^*_{ik}, P_k, r_{ik}) > 0$, we obtain a negative λ in Eq.3.23 that contradicts with the defined constraint $\lambda \geq 0$. Therefore, $\lambda = 0$ and Eq.3.22 slacks which means the service provider should not provide the capacity equal to satisfaction of his promised quality, it has to be more. Any assigned capacity can be

optimal as long as the following condition holds:

$$\phi^* = \frac{1}{Q_k} + l + \epsilon \tag{3.24}$$

ϵ represents a very small amount. By solving Eq.3.21 we can get the optimal price as follows:

$$\frac{\partial[(P_k - \phi C_k)D_i(x^*_{ik}, P_k, r_{ik})]}{\partial P_k} =$$
$$(-\frac{\alpha_i}{\gamma_i - 1})B_i r_{ik}^{\beta_i(\frac{-\gamma_i+1}{\gamma_i-1})}\gamma_i^{\frac{-\gamma_i+1}{\gamma_i-1}}P_k^{\frac{-\alpha_i}{\gamma_i-1}-1} -$$
$$\phi C_k(\frac{-\alpha_i}{\gamma_i - 1} - 1)B_i r_{ik}^{\beta_i(\frac{-\gamma_i+1}{\gamma_i-1})}\gamma_i^{\frac{-\gamma_i+1}{\gamma_i-1}}P_k^{\frac{-\alpha_i}{\gamma_i-1}-2} = 0$$

$$P^*_k = \phi C_k(\frac{\alpha_i + \gamma_i}{\alpha_i + \gamma_i - 1}) \tag{3.25}$$

Obtaining optimal response points enables us to develop an equilibrium algorithm to solve our proposed Stackelberg game. According to Algorithm 3.1, the utility of predicted rating is calculated for the service provider, then the user demand is calculated and the final provider payoff is obtained.

3.6 Simulation Results and Analysis

In order to evaluate our proposed Stackelberg game, we performed our experiments on three real life cases. We chose HostGator, Carbonite, and AceHost as our Stackelberg leaders. They all are actual IaaS providers who offer cloud backup and hosting services to business and individual users. The intuition behind selecting these three providers was their difference in average rating values that make each of them in high, middle, and low class of ratings. This section provides the simulation of users' demands and

Algorithm 3.1 PP/UP Stackelberg Game

procedure INPUT: Set $i = 1, 2, ..., n$; $\alpha_i > 0$; $0 < \gamma_i, \beta_i < 1$; Get $C_k, C_{0k}, r_{ik}, \bar{r}_k$ for service k.
  $TotalR, sum1, sum2 \leftarrow 0$
  for each $i \in N$ **do**
    Predict the rating ▷ use Eq.3.7
    if $r_{ik} \geq \bar{r}_k$ **then**
      $R_{ik} \leftarrow r_{ik}$
    else
      $R_{ik} \leftarrow -r_{ik}$
    end if
    $TotalR \leftarrow R_{ik} + TotalR$
    Obtain the optimal P_k ▷ use Eq.3.25
    Calculate x_{ik} ▷ use Eq.3.18
    Calculate D_i ▷ use Eq.3.1
    Obtain the optimal ϕ ▷ use Eq.3.24
    $UP_i \leftarrow D_i - P_k$
    $sum1 \leftarrow sum1 + P_k * D_i$
    $sum2 \leftarrow sum2 + \phi * C_k * D_i$
  end for
  $PP \leftarrow sum1 - sum2 + TotalR - C_{0k}$
end procedure

assesses how the users react to changes in the price, rating, and volume of each service. It helps investigate how the profit obtained by service providers in each rating class varies when the user sensitivities towards the service volume, rating and price change.

3.6.1 Experiment Setup

As the main purpose of this experiment is to demonstrate the reliability of the proposed Stackelberg game and its solution algorithm, we have to set meaningful data and reasonable game parameters. To do so, we obtained real world data and investigated some properties of the Cobb-Douglas function originally used in supply chain practices [98]. We simulated 300 cloud service users for each of the providers using real customer ratings from the Trust Feedback Dataset, provided by Noor et. al. [64] in the Cloud Armor project, with respect to speed and response time. HostGator has a very good record of user ratings with an average of "4.72". Afterwards, Carbonite has an average record of user ratings "2.58", while AceHost has

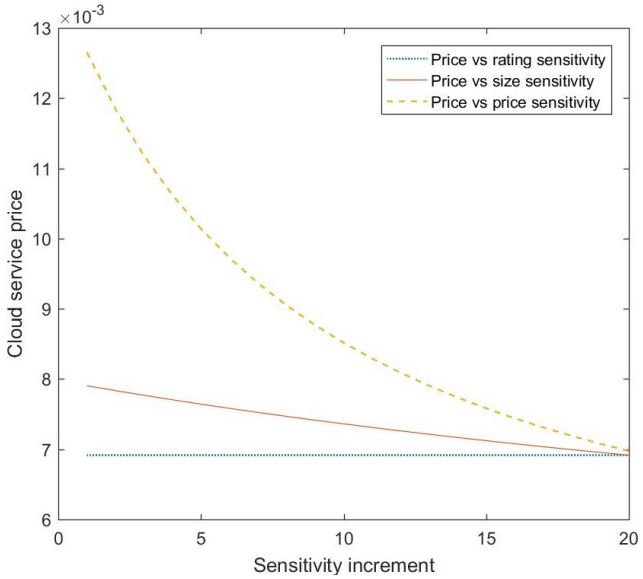


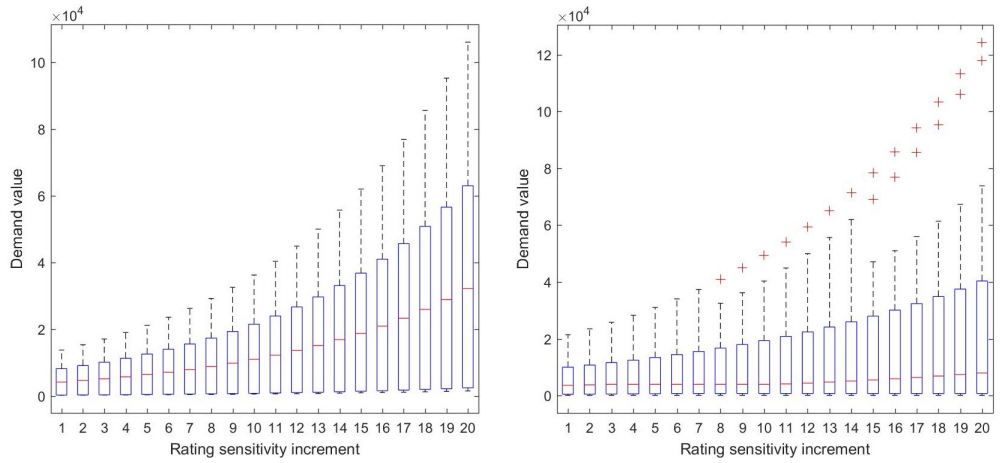
Figure 3.1: Pricing strategies (AceHost)

low record of user ratings "1.83".

Considering the fact that users usually rate the price according to their budgets, we scaled up the daily budget of users based on their ratings given to the service price factor. To obtain the process rate, we referred to the providers promised quality in the SLA statements. For example, Carbonite promises the minimum speed of 2 mbps, and from this value we computed the process rate ϕ for a day with l= 5 requests per second that gives a reasonable response time of 0.01. We set the constant scale of μ to 1 consistently with previous literature [40]. Since there is no information available about the providers' cost, we assume that they are renting their cloud infrastructure from Google, so the margin cost is obtained from Google Cloud Storage that is $C_k = 0.026$ monthly.

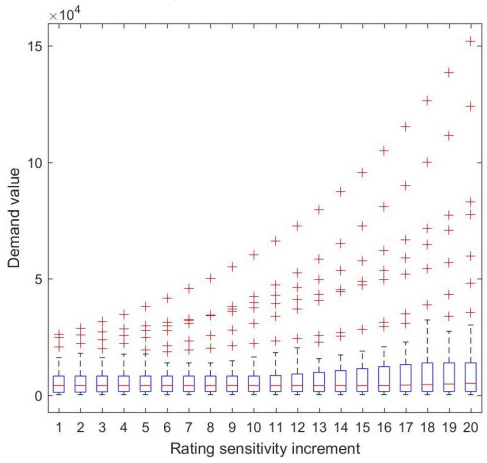
3.6.2 Rating Prediction

Through the dataset, we tried to find similar services that had ratings for the same quality factor. We identified 14 well-known service providers such as Go Daddy,



a) HostGator

b) Carbonite



c) AceHost

Figure 3.2: Demand variation with rating elasticity (20 different values of β [0.25-0.61]

Dropbox, and Dream host including the previously three nominated service providers who offer similar services. The rating prediction was conducted with a Mean Absolute Error of 1.209, and Root Mean Square Error of 1.478.

3.6.3 Pricing Strategies

Service provider has to set the optimal price based on the predicted user demand response given the offered price. Considering possible users reactions towards the given price can help service provider as a leader to choose the best pricing strategy.

This reactions towards changes in demand related parameters are to be analyzed with the defined elasticities that represents users sensitivities by changing each parameter. As an example,

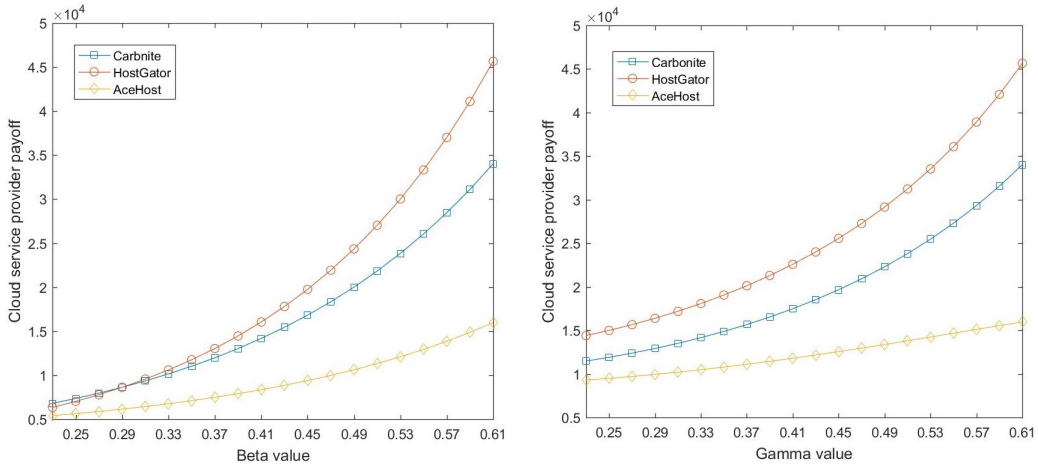
$\beta_i = \frac{\partial D_i(x_{ik},P_k,r_{ik})}{\partial r_{ik}} \frac{r_{ik}}{D_i(x_{ik},P_k,r_{ik})}$

indicates that one percentage change in r_{ik} brings a β_i percentage change in $D_i(x_{ik}, P_k, r_{ik})$.

Figure 3.1 depicts the best pricing strategies that a service providers can adopt. It is not surprising that user rating sensitivity does not affect the optimal service pricing, as it was found earlier in Eq.3.25. Meanwhile, price reduction towards size sensitivity has to be much less than what it has to be against price sensitivity. Since the optimal pricing strategies of all the three providers are similar and only differ in price reduction scale, we only provide the figure for AceHost. From these pricing strategies, we need to investigate how the users react in their demands and how these strategies will ultimately enhance the provider's profit.

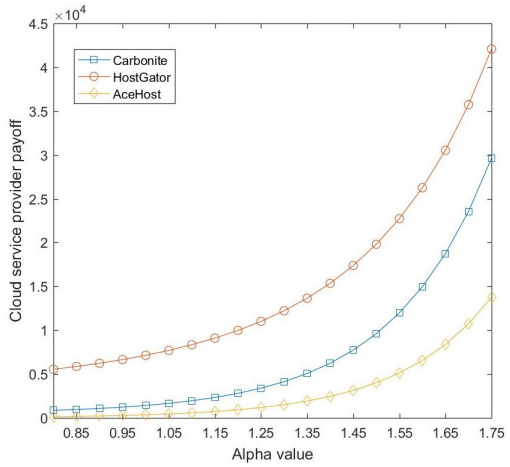
3.6.4 Sensitivity Analysis of Rating

Let us consider the rating elasticity parameter β. What a service provider in our Stackelberg game needs to know is how users will respond to ratings improvement, and how this response ultimately affects the provider's profit. In order to illustrate variations of demands within the user population, box plots are provided. Figure 3.2 shows that users' demands of all three providers rise with increase of β, but not in the same distribution. The quartiles and median of the HostGator service demand are increasing along with the growth of β. For Carbonite, the quartiles are increasing but the median remains almost unchanged. This shows less users have increased their demands. However, those who enlarged their demands, had more variation than the users of HostGator, whose variation is going up more than 120,000. AceHost has



a) User rating sensitivity

b) Service volume sensitivity



c) Service price sensitivity

Figure 3.3: Analysis of providers profit with different user sensitivities

a different situation. The majority of users' demands are unchanged, while some had increased in even more amount compared to the other two providers (more than 150,000).

The effect of these changes are reflected in Figure 3.3a, where the profits of the three providers are compared. Since the process rate and marginal cost of HostGator are high, at first Carbonite is better off. But after increasing β, HostGator outperforms Carbonite. As it was expected for AceHost, the profit has a slight improvement when β is increased.

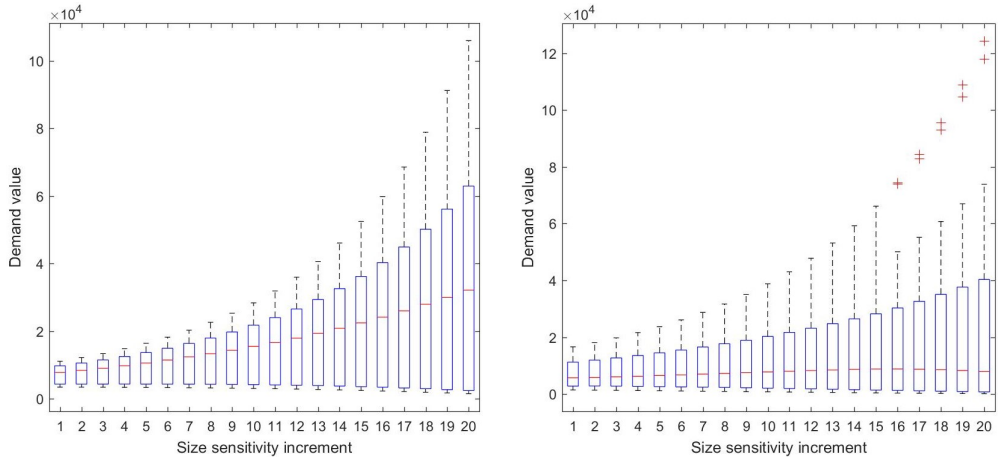
By analyzing these results, we can conclude that users who become customers of HostGator, mainly care about quality and rating. So when the provider increases his rating, he will see a dramatic increase of profit but not early. Carbonite has almost the same situation but less intense, so this provider can witness the increase of profit at slower pace. Meanwhile, the users of AceHost are not much sensitive towards rating. Therefore, rising the rating has a minor effect on AceHost profit.

3.6.5 Sensitivity Analysis of Service Volume

To estimate the volume of service that users obtain, we analyzed γ. According to Figure 3.4, variation of user demand distribution for service volume is almost the same as ratings. However, very few users have lowered their demand when they met their budget limits. Figure 3.3b shows the three providers' profit gained at a milder slope in comparison with rating increment. This is due to the fact that only few customers lowered their demands, specially HostGator' customers who should pay more money. Yet, HostGator and Carbonite have similar trend of gaining the profit out of size increment.

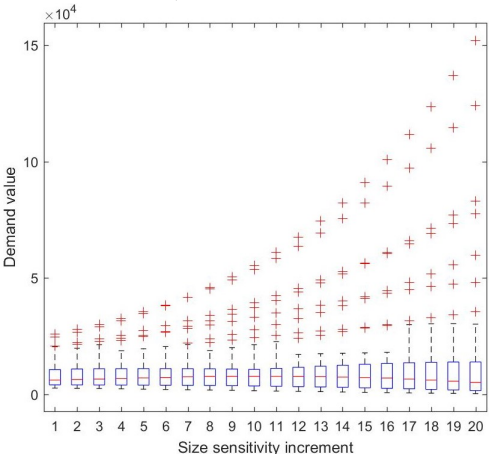
3.6.6 Sensitivity Analysis of Price

Users react differently towards the decrease of price. As α goes up, the price goes down. The change of price has to be greater than the other parameters to enhance the user demands. Figure 3.5 depicts the fact that users have a late reaction towards the decrease of price, but when they start to boost their demand, it goes up very fast. Consequently, the three providers' profits are more curvy with variation of price than the other parameters, as presented in Figure 3.3c. Like the case of the other two parameters, AceHost received less increment but most intense in variation. HostGator,



a) HostGator

b) Carbonite

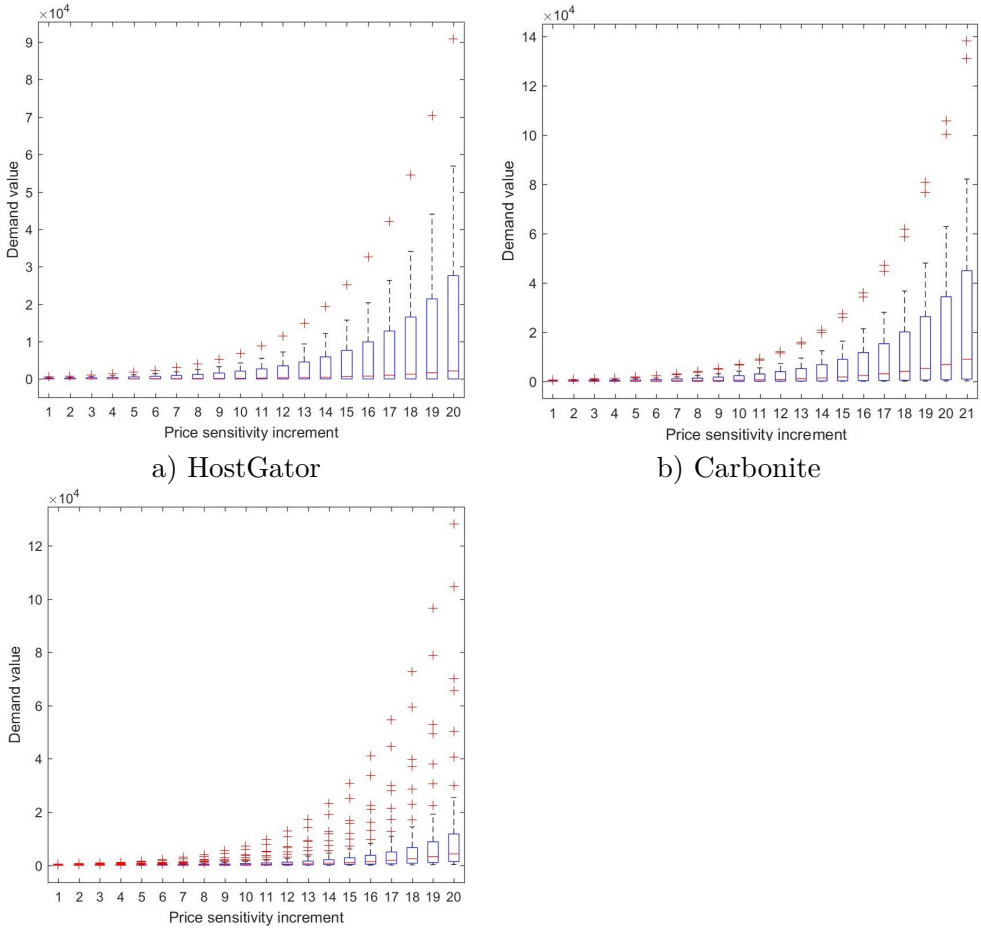


c) AceHost

Figure 3.4: Demand variation with size elasticity (20 different values of γ [0.25-0.61]

Carbonite and AceHost behave similarly at the beginning, but Carbonite profit speeds up over scaling the price reduction. This shows that medium rated providers with medium cost and price have better opportunity to gain user satisfaction by cutting the service price.

In summary, it can be inferred that users react to small changes of rating and service size, meanwhile price deduction has to be large to affect considerably the users demand. For providers with higher capacity and higher rating values, the slope of profit increment will be higher than those with less capacity and lower rating values. Although providers with high capacity and rating obtain higher profits, providers



a) HostGator b) Carbonite

c) AceHost

Figure 3.5: Demand variation with price elasticity (20 different values of α [0.85-0.75]

with low capacity and low rating may receive some unexpected demand growth by enlarging the service size, improving the rating values, or reducing the price.

3.7 Conclusion

This chapter introduced a Stackelberg game model between a typical IaaS provider and the users to optimize the profit of the service provider who operates within an on-line rating platform. The theoretically obtained results confirmed by the game simulation on a real world dataset showed that rating improvement is mostly

influential for high rated providers who compete with high quality providers and attracted the users who prioritize quality in their decision making. Improving the ratings of a low rated provider does not increase his profit as much as it does for a medium and high rated provider. Meanwhile, an average rated provider takes the most advantage out of the price reduction, that can be related to his medium cost and process rate. Lowering the price boosted almost all the users demands greatly, but only when it is reduced in large scale. In a nutshell, providers with higher capacity, rating and also cost can make more profit when the user demands increase. The main competitive advantage of high rated providers is their service quality that becomes most profitable by enhancing their ratings. Providers with lower capacity, cost and rating may see some unexpected increase of demand from some customers, but in total they will have less demand and less profit. Yet their main advantage is lower cost that attracts low budget customers with continuing their price reduction. Finally, as the competition among providers is not considered in this chapter, we design a dynamic game that models this competition over time in the next chapter.

Chapter 4

Two-Stage Game Theoretical Framework for IaaS Market Share Dynamics

In this chapter, we consider the problem of cloud market share among Infrastructure as a Service (IaaS) providers in a competitive setting. The public cloud market is dominated by few large providers, which prevents a healthy competition that would benefit the end-users. We argue that to make the cloud market more competitive, new providers, even small ones, should be able to inter this market and find a share. This problem of deeply analyzing the cloud market and providing new players with mechanisms allowing them to have a market share has not been addressed yet. In fact, to make the cloud market open and increase the cloud service demand, we show in this paper that the cloud providers have to compete not only over price, but also quality. Most of the research performed in the cloud market competition focus only on pricing mechanisms, neglecting thus the cloud service quality and user's

satisfaction. However, to be consistent with the new era of cloud computing, Cloud 2.0, providers have to focus on providing value to businesses and offer higher quality services. As a solution to the aforementioned problem, we propose a conceptual, user-centric game theoretical framework that includes a two-stage game: 1) to capture the user demand preferences (optimal capacity and price), a Stackelberg game is used where IaaS providers are leaders and IaaS users are followers; and 2) to enhance the service ratings given by users in order to improve the provider position in the market and increase the future users' demand, a differential game is proposed, which allows IaaS providers to compete over service quality (e.g., QoS, scalability and adding extra features). The proposed two-stage game model allows the new IaaS providers, even if they are small, to have a share in the market and increase user's satisfaction through providing high quality and added-value services. To validate the theoretical analysis, experimental results are conducted using a real-world cloud service quality feedback, collected by the CloudArmor project. This research reveals that due to the fact that service customization tends to enhance the customers loyalty in today's subscription cloud economy, the best strategy for small IaaS providers is to increase the service cost and improve the quality of their added-value solutions to prevent customers' defection. This not only elevates the provider's profit, but also increases the quality equilibrium that leads to a higher user satisfaction. Consequently, higher satisfaction enhances the provider's rating and future users demand. This chapter is published in [81].

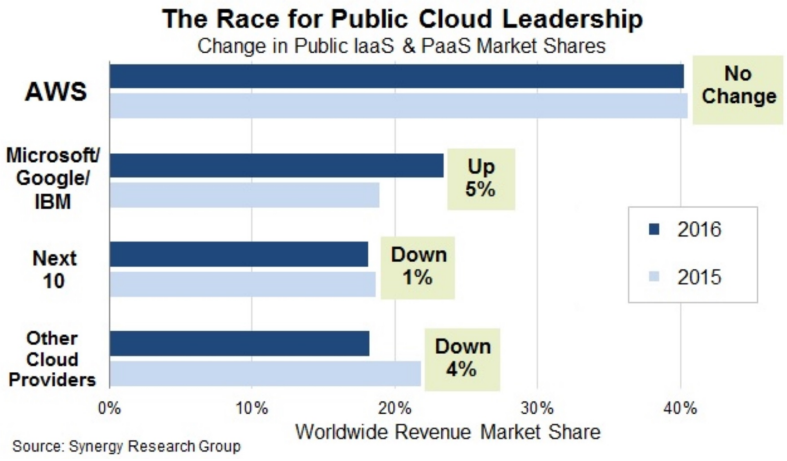


Figure 4.1: Cloud revenue race among IaaS providers

4.1 Introduction

4.1.1 Motivations

The rising demand in the cloud infrastructure service market has tempted a large number of technology providers to participate and compete in the market [15]. However, today's cloud market is dominated by only few large providers. As reported by the Synergy Research Group 2017[1], Amazon, Microsoft, Google, and IBM gained ground in the market at the expense of smaller IaaS providers. The medium sized IaaS providers lost 1% of the market and a large number of small IaaS providers collective market share dropped by 4%, as illustrated in Figure 4.1. Such a dominated market prevents a healthy competition. It also hinders compatibility with private clouds and prevents offering personalized added-value services by resellers [7]. Lack of these services may threaten the wide adoption of cloud computing in many industries. Thus, for the growth of the cloud computing industry, there is an increasing need to open the market to the new and smaller providers and create a more competitive environment.

Cloud IaaS has been in the center of attention for years and several research

[1] `www.srgresearch.com`

proposals about the technology itself have been lunched. Nonetheless, there is an urgent need to explore and address the business issues surrounding cloud computing, while considering the technical characteristics of such a paradigm. Nowadays, online market and rating platforms made it easy for users to compare a wide range of infrastructure services and for IaaS providers to establish their own credibility. In this paper, we argue that each IaaS provider entering the market needs to distinguish itself from the already established players and compete over both price and quality. As outlined in [21], today's market of Cloud 1.0 is price-focused. For that reason, there are extensive research that considered pricing competition and proposed optimal pricing strategies in order to maximize the final revenue of cloud providers [91, 95]. However, there is a large number of modern business applications for which a price-focused service model will not be adequate. Often, users hesitate to move their critical business process to the cloud since the first-generation cloud obscured its operations detail behind its low pricing models [21]. Hiding the details blurs the vision of customers about the trade-offs that the IaaS provider has made in order to offer computing at such a low price.

The new era of cloud computing, Cloud 2.0, has been emerged to focus on providing value to small and medium enterprises (SME) as well as large enterprise markets at higher costs as well as higher quality [21,34,61,74,90]. For the revolution of Cloud 2.0 to take place for IaaS, two transformations need to occur: 1) IaaS providers must be prepared to provide value to businesses that entices them out of their built-in IT resources and applications; and 2) customers must demand a combination of fast, secure, and reliable IaaS from the providers to meet their end users' expectations [13]. In fact, data security and privacy are highly important in the context of Cloud 2.0 where cloud, fog and IoT must be consolidated and application providers are granted privileges to use and process the data [73]. In this context, to ensure the availability

and delivery of low-latency services, Cloud 2.0 can be integrated with fog and edge computing to deal with the massive data volumes being produced by devices and users [75].

As an example of a cloud provider moving towards this revolution, SITA[2] is an IaaS provider that offers mobility-friendly on-demand hosting and application services specifically designed for the air transport industry. SITA has connected more than 160 airports which enabled the organization to host applications accessing to airports systems, such as terminals, gates and parking. A research conducted by Microsoft Cloud and Hosting Study[3] also confirmed the Cloud 2.0 movement by showing that 89% of companies are willing to pay additional fees for cloud management services. Despite the large number of pricing competition models, to the best of our knowledge, no one tackled the issue of the cloud providers competition from the perspective of service quality and end-users satisfaction. The only study about quality competition has been conducted by Fan et al. [24] who considered market competition among a software as a service provider and a traditional software provider. Their research focus on marketing advantages of bundling software in a service, neglecting the tight competition among cloud providers themselves and the user satisfaction effect on providers' revenue.

Considering the initiatives of Cloud 2.0 movement, this study promotes a healthy market competition through rigor economical and theoretical models. To build a practical roadmap, we propose to empower new and small providers by considering two key features of Cloud 2.0:

1. High quality services: Considering the increasing number of clouds deployed in private data centers, the classic approach, such as the one used by Amazon, to build a cloud in which hardware and software developments are insourced,

[2] https://www.sita.aero/

[3] http://partner-l1.microsoft.com/hosting-cloud-research-report-2017

is no longer efficient and hardly deployable. Instead, clouds are being built out of commercial technology stacks with the aim of enabling the infrastructure providers to access the market rapidly and compete while providing high-quality services. However, finding cost-efficient component technologies offering high reliability, continues support, adequate quality, and easy integration is highly challenging. Unlike most of the research performed in the cloud market competition focusing only on pricing mechanisms, we model the competition from the perspectives of cloud service quality and user's satisfaction by focusing on added-value and superior quality services. Enabling small or new providers to access the market and offer personalized added-value services within our proposed model is part of this feature of Cloud 2.0 that enhances compatibility with private clouds.

2. Long-term commitment: The success of modern business applications relies on the reliability of services, such as incident response, security hardening, SLA assurance, software updates, and performance tuning. In fact, 80 percent of downtime is caused by service provisioning problems. Traditionally, these services have been delivered by the IT departments, and simply deploying remote servers in the cloud doesn't solve the services problem. Because services in the cloud will most likely be outsourced, they must be delivered while considering the customer's needs in a long-term commitment vision. Moving toward this long-term commitment strategy will drive providers to better focus on customer satisfaction to enjoy higher benefits. Our simulations also confirmed that providing added-value services along with customization could increase long-term commitment which is indeed very profitable.

4.1.2 Problem Statement and Contributions

In this thesis chapter, we consider the problem of IaaS cloud market share taking into consideration the need for new cloud providers to be in the market and the requirements of Cloud 2.0. We propose a conceptual, user-centric two-stage game theoretical framework that can help the IaaS providers and users optimize the service quality with a balanced profit. The first stage of our conceptual framework uses our Stackelberg game [79] to identify the user demand preferences and set the optimal price and capacity for the IaaS provider. The Stackelberg game model focuses on interaction among a single IaaS provider with a group of users to appropriately capture the demand elasticities and set the price and allocate resources for each Virtual Machine (VM) to meet the Service Level Agreement (SLA) and match the customers interests. However, our Stackelberg model does not consider the competition among the IaaS providers to provide higher quality services. Therefore, in the second stage, we formulate this competition through a differential game with service quality features as the main competitive factors. Most of the studies on strategic interactions among the cloud participants are grounded in static frameworks [62, 101]. These models overlook the strategic issues arose when providers interact repeatedly over time. Thus, to tackle the limitation of static frameworks, we introduce a non-cooperative dynamic differential game that captures the important dimension of time.

The designed differential game takes multi-tenancy property into account, which leads to define competitive advantages for both the large and small IaaS providers. The large providers (the market leaders) make their profit through a virtuous cycle reflected through the following causal associations: 1) the more customers an IaaS provider gets, the more infrastructure and the better resource provision with robust cloud features (e.g., higher availability and more storage) it can afford; 2) the more infrastructure, the better economies of scale and the cheaper prices for IaaS; and 3)

the lower prices and the better their quality, the more customers the provider can get. Meanwhile, the small IaaS providers have fewer users and limited resources. Thus, by targeting a specific industry or local region, they can have tenants who share the same scheme with similar requirements such as complying with data and security regulations, national and international standards or dealing with compatibility issues. This enables them amalgamate their needs by customizing their services to add value to the users' business solutions. Providing personalized cloud services can further drive customer loyalty [19]. To reflect the above arguments and take them into account, we introduce three main competition factors including ratings by users that reflect customers satisfaction, low cost QoS provisioning, and customization or added-value services.

In summary, our main contribution is a two-stage game theoretical model that:

- Allows new and small IaaS providers to compete against the existing and large ones and have a market share, which enables a productive cloud market industry that benefits the end-users. To the best of our knowledge, our work is the first that investigates this competition in the cloud computing context.

- Maximizes users satisfaction modeled using users' ratings by providing a continues service quality development. It is the first research that models a dynamic competition considering the quality of service among IaaS providers.

- Captures user preferences and demand elasticities for optimal price and resource allocation. To ensure the continued validity of the optimality in the presence of changing internal or external factors, a post-optimality analysis is provided.

The proposed model can help new born IaaS providers identify their users' needs and potential markets, anticipate their competitive advantage, formulate their valuation model and create new service provisioning scenarios. We implement our model using

a real-world dataset containing users' ratings over cloud service quality features, obtained from the CloudArmor project[4]. Finally, it is worth mentioning that because the problem of making the cloud market competitive by analyzing how small providers can get a market share has not been addressed yet, no benchmark has been found for the purpose of comparison.

4.2 Related Work

Small and medium businesses can take the advantage of cloud computing in several ways [72]. Cloud computing offers scalable services that businesses can use on demand as much as they need to. The competitive market of cloud services provides a variety of options in pricing and quality. The users can always shift their host provider to another provider offering more opportunistic service or lower price. Due to this opportunistic characteristics, this industry is predicted to reach $270 billion in 2020 [97]. Cloud economics plays a significant role in shaping the future of cloud computing industry. The economics of the cloud computing can have two dimensions [65]: 1) intra-organization that deals with internal factors such as labor, power, hardware and so on; and 2) inter-organization that refers to market competition factors between organizations such as price, quality of service, and reputation. A third dimension can also be considered where providers can adopt a cooperation strategy by forming coalitions or federations among data centers [87]. In such federations, different challenging problems have been addressed including virtual network provisioning [76] and trust management [88]. This study deals with the second dimension. In this section, we present the work related to market share modeling from economics and marketing literature followed by the work done related to cloud services quality and

[4] `https://cs.adelaide.edu.au/~cloudarmor/ds.html`

pricing strategies.

4.2.1 Market Share Dynamics

Most of the proposals in the literature about market share are static [33]. A non-static approach has been taken by Breton et al. [10], who studied dynamic equilibrium advertising strategies in a duopoly market. They defined a model to formulate the market share dynamics for two competitors and obtained a feedback differential Stackelberg equilibrium. Gutierrez et al. [30] analyzed the dynamic strategic interactions between a manufacturer and a retailer in a distribution channel for innovative products. The underlying assumption was that the retail demand for such a product is influenced by word-of-mouth from past adopters. This influence creates a trade-off between immediate and future sales and profits of the manufacturer. The obtained equilibrium dynamic pricing showed that in some cases, far-sighted retailer is more profitable. The above mentioned studies utilize differential game to help businesses optimize their sale and advertisement channels regardless of the customer satisfaction, while this paper considers the technical characteristics of infrastructure cloud computing environment to distribute a fair market share among IaaS providers and fulfill the users' requirements.

Only few proposals have explored the users' ratings impact on business owners profit [16]. Nonetheless, their importance in marketing strategies has been recognized [80]. Duan et al. [20] studied video sales and movie recommender systems and found that users' ratings reflect movies quality, but they do not persuade the users to buy. In fact, they increase the users' awareness by word-of-mouth that is central to the efficacy of providers and increases their sales directly. Completing their study, our research proves that cloud service quality significantly affects the overall users' ratings, and further shows how cloud providers can take the advantage of those ratings to enhance

reputation and increase profit.

4.2.2 The Competition among Cloud Service Participants

Game theory has been successfully applied in the cloud computing area, for instance for resource allocation and pricing mechanisms, where the interactions of players have to be taken into account [65]. A user-provider interactive approach is taken by Hadji et al. [31], where a Stackelberg game is designed to consider constrained pricing with limited resources offered by a cloud service provider and the optimal user demands. Xu et al. [91] optimized a pricing policy for cloud service providers to better compete with each other under the evolution of the cloud market. Forming a Stackelberg game, the authors applied a reinforcement learning (Q-learning) to find out an optimal policy for the leader provider. Following the leader, the optimal policy for followers will be uncovered. In the same line of research, Shen et al. [71] used a Stackelberg game to model the interactions among data providers, service providers, and users. The authors studied the optimization problem of the players' profits using deep learning in a context of data markets. However, price is the only utility factor considered in these studies and the importance of QoS and user satisfaction is somehow neglected.

Zhao et al. [103] investigated the impact of the two factors of energy consumption as well as SLA violations on degrading the cost-efficiency of data centers and the cloud providers' revenue. The authors developed online VM placement algorithms as an optimization problem of maximizing revenue from VM migration and achieved promissing results. The research conducted by Kilcioglu et al. [42] calibrated a static model for price-quality trade-off in two cases of monopoly and duopoly price competitions where the IaaS marketplace is referred to as commoditized from the perspective of economic competition. The reason is that cloud providers use similar physical hardware which cannot be differentiated from each other and profit margins

should become null. The conducted experiment explained the price cutting behavior of the current market trend and also how providers are able to make a profit despite predictions that the market should be totally commoditized. Conversely, this paper emphasizes a different approach aligned with the vision of Cloud 2.0. Commoditization for young and small competitors is not profitable and these providers cannot survive in the market of Cloud 2.0 due to their lower number of users and higher expenses. We advocate smaller providers to differentiate themselves from the established large providers in the market by providing added-value services to their customers.

The only study on cloud service quality that inspired our research is performed by Fan et al. [24] who considered market competition among a software as a service provider and a traditional software provider as a differential game. This research analyzes a short and long-term competition for price and dynamic quality between the two firms. The authors found that the cost of software implementation can significantly affect the equilibrium price while quality improvement has a more robust effect. Our work differs from this research in many points: 1) we focus on internal competition among IaaS providers considering the technical advantages and challenges specific to IaaS, specifically when a new provider enters the market to compete with big and dominant providers; 2) the user demand is formulated based on the user preferences and the two proposed game models prioritize the user satisfaction considering price, capacity and quality optimization; and 3) our model contains a continuous game loop where the players enter two different games and can evaluate post-optimality analysis to choose the right game, the right stage, and the right time to enter and to stay. Our post-optimality analysis also informs the players about the changes to the optimum values as they change over time.

4.3 Framework Overview

The race to maximize the revenue, specifically for the new entrants to the cloud market, entails formulation of non-cooperative games. We form two key competing players representing each a group of the same type: 1) a small and fresh provider, and 2) a large and reputed provider. Early game theoretic models in product competition emphasized static models. A dynamic model can add the important dimension of time and recognizes the competitive decisions that do not necessarily remain fixed. Models involving competition in continuous time are typically treated as differential games, in which critical state variables, e.g., demand or market share, are assumed to change with respect to time according to specified differential equations. Differential games have been widely applied in various domains to analyze competition in dynamic advertising and pricing [30].

This research tackles the problem of maximizing the IaaS providers' revenue through two interactive games in a cycle with 11 steps, as presented in Figure 4.2. The first stage is the cloud market identification and demand provisioning for a new IaaS provider. The box on the top including the service selection Stackelberg game illustrates six interactive steps among an IaaS provider as the leader and the IaaS users as the followers. In the first and second steps, the IaaS provider k_J announces its price, quality and the average rating obtained so far. Then, the users decide the amount of their request (step (3)). The IaaS provider predicts the future user rating, plans the optimal capacity and offers the actual price under guaranteed SLA (steps (4 and 5)). In the final step (6), the user provides its rating. A brief explanation of this model needed in the rest of this report is provided in the next section and more details can be found in our previous work [79]. This game produces two outcomes: optimum price and capacity of VM (P^*, ϕ^*).

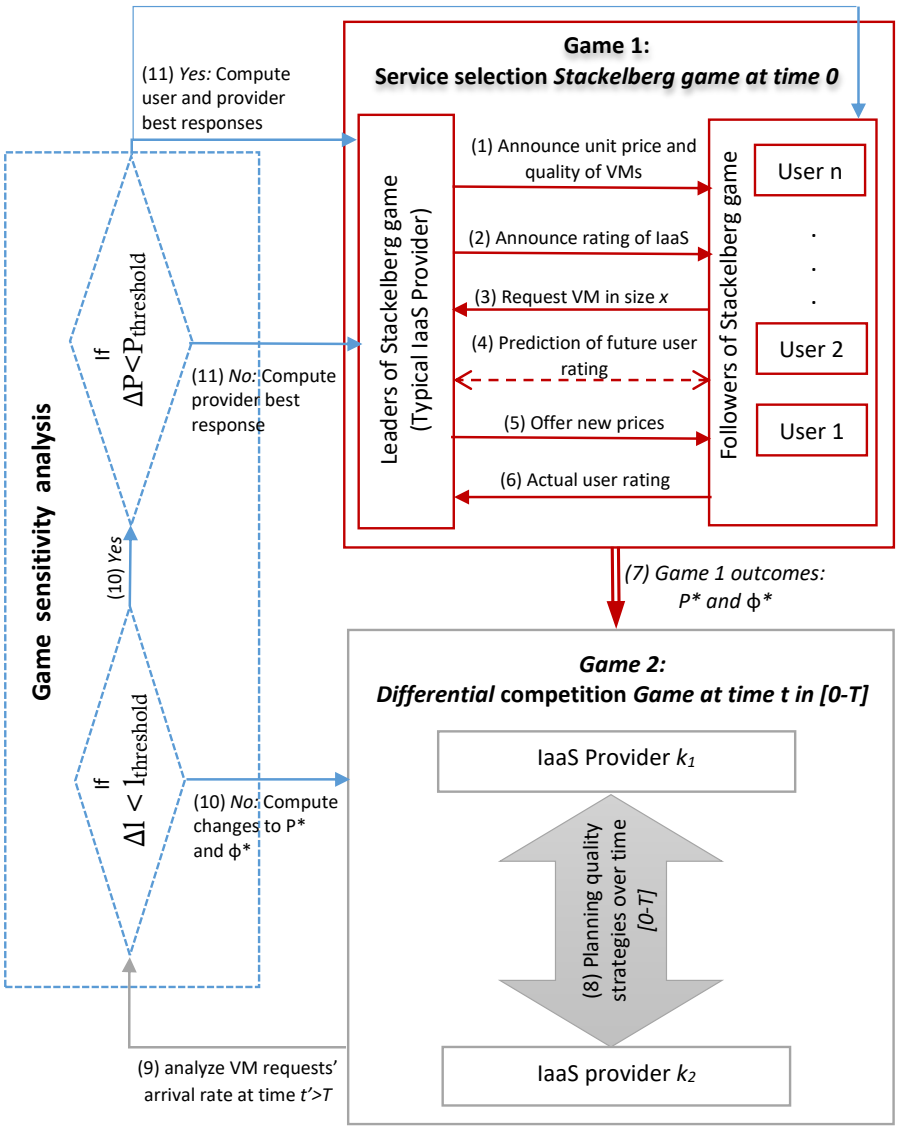


Figure 4.2: Hierarchical Stackelberg and differential games' framework

After setting the price and capacity (step (7)), the provider enters into the second game (called differential competition game) which is proposed in this report. During step (8), the IaaS provider has to compete with all the existing IaaS providers to enhance its service ratings through a justified amount of quality increments. The IaaS quality factors include functional and non-functional attributes such as QoS, adding new features, scalability and security. As our objective is to analyze the entrance of

new providers to the cloud market, we denote by k_1 a typical small and new IaaS provider, and by k_2 a typical well-established IaaS provider competing against k_1. The outcome of the differential game (Game 2) is the required amount of quality improvement during the time interval $[0 - T]$ for a given T.

In the meantime, the optimality of the obtained values has to be analyzed since the game is dynamic and the values of the variables are changing. The users request (in terms of VM) arrival rate l that depends directly on the number of end users, will be used to assess the optimality of VM's price and capacity in step (9). Thus, if the variation of l remains less than a threshold, no changes are required and players shall stay in the second game (step (10): No). However, if the variation exceeds the threshold, a new optimal price has to be calculated using the new value of l (step (10): Yes). In the event that the price deviates from a certain threshold (the game sensitivity analysis), the game players have to go back to the Stackelberg game (Game 1) and start over the game, which includes computing the provider and users' best responses (step (11): Yes). Otherwise, the two IaaS providers only need to recompute their own best responses and obtain a new price and capacity through Game 1 (step (11): No).

4.4 IaaS Selection Stackelberg Game

In the first game, we model the cloud service market interactions between a typical IaaS provider and the service users as a Stackelberg game, where the IaaS provider is the leader and the service users are the followers. The users observe the price and ratings to adjust their demand accordingly. In quest of the users' demands, the IaaS provider makes a decision on its pricing strategy and optimal capacity. The game parameters are provided in Table 4.1. In order to capture demand elasticities and

Table 4.1: Notations used in the service selection Stackelberg game

Decision variables	
x_i	VM request size of user i for the IaaS
ϕ	VM preserved capacity
P	Price per VM of the IaaS
Input parameters	
$i = 1, 2, ...n \in N$	Index of n users in the set N
B_i	User i budget
R_i	Rating utility of IaaS provider from user i
r_i	Service rating of user i for the IaaS provider
α_i	IaaS price elasticity for user i
β_i	IaaS rating elasticity for user i
γ_i	Size and number of VMs elasticity for user i
l	VM requests' arrival rate
μ	Constant scale of user IaaS demand
Q	Guaranteed QoS as stated in SLA
C_0/C	Fixed/marginal cost of the infrastructure
$\lambda_{i1}/\lambda_{i2}/\lambda_k$	Lagrange multipliers

variations specific for each user, we defined the user demand using the Cobb-Douglas function that models well these elasticity aspects in terms of price and rating [29]. It is assumed that the user will have the opportunity to check the IaaS provider rating that represents the actual user satisfaction level. The user demand function is defined as follows:

$$D_i(x_i, P, r_i) = \mu \; x_i^{\gamma_i} \; P^{-\alpha_i} \; r_i^{\beta_i} \tag{4.1}$$

where α_i, β_i and γ_i, $i = 1, 2, ..., n$ are elasticities (variations) of the IaaS price P, rating r_i and VM size x_i respectively. The price elasticity α_i is dependent on the user i because it reflects the price the user is willing to pay for the provided infrastructure. Different market users, having different requirements and satisfaction levels, do not

react evenly to the same price or rating. It is the combination of these factors that produces different values of α_i, β_i and γ_i. The user aims to maximize its payoff as follows:

$$\begin{aligned} &\text{maximize} \quad UP(x_i) = D_i(x_i, P, r_i) - P \\ &\text{subject to} \quad Px_i \leq B_i, x_i \geq 0 \end{aligned} \tag{4.2}$$

As the IaaS provider needs to maintain its reputation through the user ratings, it is responsible to process users requests on time. Thus, it is important to consider VMs processing rate that represents VM capacity of the IaaS provider, denoted as ϕ. A large processing rate requires increasing the number and capacity of VMs, meaning a higher cost for ϕ that includes fixed cost of C_0 and marginal cost of C. Thus, the total cost for VM capacity ϕ is $C_0 + C\phi$.

Following previous literature in cloud computing [24], we model the arrival of VM requests as a Poisson process with mean arrival rate l. The average delay for a request in an M/M/1 queue can be defined as $\frac{1}{\phi - l}$. The IaaS provider aims to optimize its profit by increasing the price and ratings given by the users, and minimizing the costs. Thus, the IaaS provider's (i.e., the leader) optimization objective is:

$$\begin{aligned} &\text{maximize} \quad PP(P, \phi) = \sum_{i=1}^{n} (P - \ \phi \ C) \ D(x_i^*, P, r_i) + \ \sum_{i=1}^{n} R_i - C_0 \\ &\text{subject to} \quad \frac{1}{\phi \ - \ l} \leq Q \\ &\qquad\qquad \phi > 0, P > 0 \end{aligned} \tag{4.3}$$

R_i is the rating utility that is affected positively if the given rating is above the average user rating, and is affected negatively if the rating is below that average. Details of user rating prediction and rating utility calculation can be found in our previous work [79]. We solve the equilibrium point of the above defined Stackelberg game by a backward induction procedure. Therefore, the followers' (users) problem has to be

solved first to obtain the response function of these users. The leader's (provider) decision problem is then computed considering all possible reactions of its followers in order to maximize its net profit.

4.4.1 User Best Response

As the objective function in Eq.4.2 is continuous and concave in x_i, we obtain the solution using Lagrange multipliers, λ_{i1} and λ_{i2}, with Kuhn-Tucker conditions. So, we will have a new objective function:

$$L_{UP} = D_i(x_i, P, r_i) - P - \lambda_{i1}(x_i P - B_i) + \lambda_{i2} x_i \tag{4.4}$$

with the following conditions:

$$\lambda_{i1}(x_i P - B_i) = 0 \tag{4.5}$$

$$\lambda_{i2} x_i = 0 \tag{4.6}$$

Where $\lambda_{i1}, \lambda_{i2}, x_i \geq 0$. The only coupling point between users is x_i, so we take the derivative with respect to x_i.

$$\frac{\partial L_{UP}}{\partial x_i} = \frac{\partial D_i(x_i, P, r_i)}{\partial x_i} - \lambda_{i1} P + \lambda_{i2} = 0 \tag{4.7}$$

We have two cases: 1) $x_i = 0$: regardless of the value of λ_{i1} and λ_{i2}, this means the user is not demanding any services, so its utility will be null; and 2) $x_i > 0$: from slackness complementary condition in Eq.4.6, we can conclude that $\lambda_{i2} = 0$; so we

have:

$$x_i = (\frac{\lambda_{i1} P^{\alpha_i+1}}{r_i^{\beta_i} \gamma_i \mu})^{\frac{1}{\gamma_i - 1}} \tag{4.8}$$

By substituting x_i from Eq.4.8 in Eq.4.5 we obtain λ_{i1}:

$$\lambda_{i1}^{\frac{1}{\gamma_i - 1}} = \frac{B_i r_i^{\beta_i} \gamma_i \mu}{P^{\frac{\alpha_i+1}{\gamma_i - 1}+1}} \tag{4.9}$$

The final response x_i from user i is attained by replacing Eq.4.9 in Eq.4.8.

$$x_i^* = \frac{B_i (r_i^{\beta_i} \gamma_i \mu)^{\frac{\gamma_i - 2}{\gamma_i - 1}}}{P} \tag{4.10}$$

The above obtained x_i^* is optimal where Eq.4.5 slacks and $\lambda_{i1} > 0$. However, we claim that it is reasonable to consider slackness rather than binding, since having $\lambda_{i1} = 0$ is an extreme case where the user cares only about the price and does not consider the previous ratings or quality.

4.4.2 IaaS Provider Best Response

Using Lagrange multiplier λ_k, we model the objective optimization in Eq.4.3 as follows:

$$L_{PP}(P, \phi, \lambda_k) = PP(P, \phi) - \lambda_k (\frac{1}{\phi - l} - Q) \tag{4.11}$$

The Kuhn-Tucker condition for our model is:

$$\frac{\partial PP(P, \phi)}{\partial \phi} - \lambda_k \frac{\partial (\frac{1}{\phi - l} - Q)}{\partial \phi} = 0 \tag{4.12}$$

$$\frac{\partial PP(P, \phi)}{\partial P} = 0 \tag{4.13}$$

$$\lambda_k(\frac{1}{\phi - l} - Q) = 0 \tag{4.14}$$

where $\lambda_k \geq 0, P, \phi > 0$. To find the optimal capacity ϕ, we first assume that Eq.4.14 binds and $\lambda_k > 0$. Referring to Eq.4.12 we have:

$$CD_i(x_i^*, P, r_i) - \lambda_k(\frac{-1}{(\phi - l)^2} = 0$$

$$\lambda_k = -CD_i(x_i^*, P, r_i)(\phi - l)^2 \tag{4.15}$$

Knowing that $C > 0$ and $D_i(x_i^*, P, r_i) > 0$, we obtain a negative λ_k in Eq.4.15 that contradicts the defined constraint $\lambda_k \geq 0$. Therefore, $\lambda_k = 0$ and Eq.4.14 slacks, which means the IaaS provider should provide a higher VM capacity than what is promised in SLA. Any assigned capacity can be optimal as long as the following condition holds:

$$\phi^* = \frac{1}{Q} + l + \epsilon \tag{4.16}$$

ϵ represents a very small amount. By solving Eq.4.13 we get the optimal price as follows:

$$P^* = \phi^* C(\frac{\alpha_i + \gamma_i}{\alpha_i + \gamma_i - 1}) \tag{4.17}$$

4.5 Differential Competition Game

In the previous section, we designed a static game between a typical IaaS provider and a typical IaaS user. However, the IaaS provider does not act alone in the market. After identification of the users' requirements, the IaaS provider needs to plan a suitable strategy against its competitors. To model dynamic competition among the

IaaS providers over a period of time, we design a differential game with continuous strategies over a finite horizon time T. The game is between a typical new and small IaaS provider k_1 with n customers, and a typical large and established IaaS provider k_2 with m customers, $m >> n$. The list of employed notations is given in Table 4.2. For the sake of simplicity, we use k when we refer to both providers. Considering the cost, technical reality and multi-tenancy characteristics of cloud computing, each IaaS provider faces different challenges to compete with high quality services. In the next section, we explain which of those can lead the way of IaaS providers.

4.5.1 IaaS Architecture and Competitive Advantage

To scale the economical benefits and optimize resource utilization, multiple VMs are initiated on the same physical server simultaneously. Multi-tenancy implies multiple customers of a services set. For instance, multiple business units within a large organization with resources and data that should remain separate through a logical segmentation of the shared infrastructure by using software-defined technologies. The segmentation options available to be considered are: physical separation, logical separation, data separation, network separation, and performance separation. In the performance separation scheme, the infrastructure is shared but the capacity or QoS is guaranteed while no other separation scheme ensures such a quality. As discussed earlier, despite the tremendous momentum of the cloud computing, many firms are reluctant to move to the cloud due to the performance concerns. For that reason, we only consider the option of "performance separation" in our model, which remains a key component in the Cloud 2.0 movement. Considering the same scheme, each provider may take its own advantage to compete.

Competitive advantages of large IaaS providers (k_2): Tenants with guaranteed

performance require consistency and predictability which are challenging for IaaS providers since infrastructure is shared by many tenants. For the sake of performance isolation, it is not enough to use host-based virtualization technologies since the bandwidth between VMs of the same tenant can change significantly over time. This variation depends on the network load and usage peak from other tenants. The larger number of customers, the less variation is expected from the overall average demand. A large IaaS provider that serves a large number of customers and operates within different industries and geographically dispersed locations can avoid the cost of overbooking. New scheduling algorithms allow multiple workloads on the same cluster of customers to access a common data pool along with hardware and software resources. Thus, larger providers can balance the workloads of the same clusters and would achieve the required performance with less preserved capacity. On the other hand, a smaller IaaS provider with fewer customers has to provide a larger amount of reserved capacity to meet the variation. Another advantage of large infrastructure is energy saving cost of data centers. Large tenant clusters enable providers to shut down the idle servers and migrate tasks to other VMs running on active servers.

The multi-tenancy architecture is not visible to the user, however the user can observe its effects. The visible effects, such as higher availability and scalability, are directly reflected in our model through user rating (r_i) that ultimately increases the user demand and provider's revenue. Further, we consider the invisible effects of multi-tenancy for the IaaS providers (ζ_{k_2}). ζ_{k_2} is mainly considered as a discounted cost for the large provider due to its larger infrastructure as explained earlier.

Competitive advantages of small IaaS providers (k_1): Security regulations vary specially when the IaaS provider has to operate in diverse national and international markets. Thus, customers require to customize the security and access settings to each

region or country's regulations. The same thing can happen with regulated industries. Mobile realm is another example where customization is highly desired. Organizations are more and more adopting mobile applications and require to integrate them into their cloud infrastructure without introducing network risks. The lack of personalized infrastructure services may not allow organizations to gain the maximum potential value of their cloud investments. Despite the importance of customization, it is disputed that large providers are not willing to offer customized service-oriented architecture or application programming interfaces to SMEs [56]. Customized use of the cloud in a multi-tenancy environment is costly and hard to realize, unless the customers residing in a cluster share the same scheme. Having similar requirements (e.g., same regulations) enables the small IaaS provider to support added-value options for each application type. Since data sources for multiple tenants are in the same database, by using a simple data aggregation, the small IaaS provider can develop applications specific for its group of customers. Taking the explained example of SITA, the provider created a large network of organizations working in the air industry and expanding their offers to applications built on top of that network.

This competitive factor is not embraced in user ratings collected through online platforms and neither relates to cost. However, it definitely has a strong positive effect on the users' demand. Thus, we use ϑ_{k_1} to reinforce the formulated demand.

4.5.2 Differential Game Formulation

IaaS providers establish their credibility and gain their share through on-line market platforms where users can express their ratings. We make the following realistic assumption that is satisfied in market platforms in general.

Assumption 4.1. The quality of an IaaS has a significant effect on users'

rating, so that improving the service quality will ultimately lead to the increase of the average users' rating.

Assumption 4.1 is the basis of our game design and is very important for the validity of our game. Thus, we will validate this assumption using statistical analysis in Section 4.7.

Conventionally, IaaS providers may compete over two factors: price and quality. Several research proposals showed that due to the cost of changing price and being inconvenient for the customers, prices do not change frequently [24]. Thus, we consider the optimum price as computed in the first game in Eq.4.17 for each provider, and we assume it remains constant over the time interval $[0 - T]$, while the quality can be updated throughout the game. Afterward, whenever the values of the defined parameters vary significantly enough to warrant a shift in price, the optimal price can easily be determined by solving the Stackelberg game with the new parameter values. Consequently, the new optimal price can be utilized to solve the differential competition game over the next period of time. The following definition explains the service quality factors featured in the game.

Definition 4.1. The quality factors of an IaaS can be any of the following elements:

- QoS: basic quality features such as response time, throughput, and availability.
- Adding new service features and innovative / customized offers to the existing service.
- Enhancing and optimizing cloud specific features such as elasticity, security, and storage space.
- Supporting customers technically or non-technically.

Table 4.2: Notations used in the differential competition game

Decision variable	
$q_k(t)$	Quality control path (quality improvement) of IaaS k at time t
Input parameters	
k_1	New and small IaaS provider
k_2	Existing and big IaaS provider
δ_k	Customers' defection rate to buy IaaS k
ρ	Discount rate of future IaaS provider's revenues
$\hat{\theta}_k$	Rate of IaaS k demands increase
$\check{\theta}_k$	Rate of IaaS k demands drop
η_k	Non-functional cost of IaaS quality (e.g. quality attributes achieved by preserving higher VM capacity)
f_k	Functional cost of IaaS quality (e.g. offering new features and improving technical support)
ζ_{k_2}	Rate of discounted non-functional cost due to large infrastructure for IaaS provider k_2
ϑ_{k_1}	Rate of customization value for customers of IaaS provider k_1
$[0-T]$	Time horizon of the game

The next assumption establishes the initial conditions to formulate our differential competition game.

Assumption 2. Each player (IaaS provider) has perfect knowledge of:

- The function $\dot{D}_k(t)$ determining the evolution of the user demand, and the control path of $q_k(t)$ available to the two players.
- The payoff function PP_k.
- The initial demand state at time zero, $D_k(0)$.

However, players have no knowledge about the future states. So, they will not be able to observe the state and update their initial control path ($q_k(t)$) of quality improvement. The information structure of the game is open-loop. This means the players must formulate their decisions at time t only with the knowledge of the initial condition of the state at time zero. The intuition behind the selection of this

information structure is that IaaS providers have to put some investment and make stable pricing strategies at the initial stage because changing these strategies bears some cost for both providers and their customers. Besides, quality improvement may result in the increase of immediate rating, but improving the average rating is a long-term strategy and is not observable in short time.

In traditional service trading models, once customers had chosen their providers, they tended to keep the relation working since the investment has been made through a long-term contract. In the subscription cloud economy, customers are much free to defect anytime from a provider and switch to another one as there is very little to no financial penalty to do so. There are several variables that affect the user's demand over time. The demand for an IaaS increases with its rating improvement. Due to the strong correlation between rating and quality as asserted in Assumption 1, we use quality instead of rating. Therefore, improvement of quality elevates the demand at a rate of $\hat{\theta}_k$. We also define a demand drop rate $\check{\theta}k$ when the other provider enhances its service quality. Moreover, customers may defect at a certain rate δ_k. Based on the predefined variables, the users' demand dynamics evolve according to the following equations:

$$\begin{cases} \dot{D}_{k_1}(t) = (\hat{\theta}_{k_1} + \vartheta_{k_1})\ q_{k_1}(t)^{\beta_n} - \check{\theta}_{k_1}\ q_{k_2}(t)^{\beta_n} - \delta_{k_1} D_{k_1}(t) \\ D_{k_1}(0) = D_{k_1 0}, 0 < t < T \end{cases} \tag{4.18}$$

$$\begin{cases} \dot{D}_{k_2}(t) = \hat{\theta}_{k_2}\ q_{k_2}(t)^{\beta_m} - \check{\theta}_{k_2}\ q_{k_1}(t)^{\beta_m} - \delta_{k_2} D_{k_2}(t) \\ D_{k_2}(0) = D_{k_2 0}, 0 < t < T \end{cases} \tag{4.19}$$

As discussed earlier, ϑ_{k_1} is the added-value to the quality for IaaS provider k_1. β_n and β_m denote the average users' sensitivity towards rating for k_1 and k_2 respectively. Eq.4.18 and Eq.4.19 explicitly describe how the service quality of the two competitors

jointly determine the dynamics of demand rate.

The marginal cost of increasing quality is considered to be quadratic in past studies [24]. We consider the same quadratic increment for the increase of quality. Thus, let $\hat{C}q_k(t)$ to be a cost associated with the efforts to increase the quality level by an amount $q_k(t)$ at time t. Two types of quality improvement are considered in our model: functional (f) and non-functional (η). The functional quality improvement is realized by adding extra functionalities to the service, such as offering new features or improving technical support. The non-functional one is related to the quality attributes that can be reached by reserving extra resources and increasing the processing capacity. Definition 2 determines the cost of quality improvement for both IaaS providers.

Definition 4.2. To increase the quality, k_1 and k_2 incur a quadratic cost function as follows:

$$\hat{C}q_{k_1}(t) = f_{k_1}q_{k_1}(t)^2 + \eta_{k_1}q_{k_1}(t)^2 \tag{4.20}$$

$$\hat{C}q_{k_2}(t) = f_{k_2}q_{k_2}(t)^2 + (\eta_{k_2} - \zeta_{k_2})q_{k_2}(t)^2 \tag{4.21}$$

Concurring with the cost functions delineated by Definition 4.2, the instant profit of IaaS providers k_1 and k_2 at time t can be calculated according to the following formulas:

$$\begin{aligned} PP_{k_1}(D_{k_1}(t), q_{k_1}(t)) \quad &= (P^*_{k_1} - \ \phi^*_{k_1}\ C_{k_1})\ D_{k_1}(t) \\ &\quad - (f_{k_1}q_{k_1}(t)^2 + \eta_{k_1}q_{k_1}(t)^2) - C_{0k_1} \end{aligned} \tag{4.22}$$

$$\begin{aligned} PP_{k_2}(D_{k_2}(t), q_{k_2}(t)) \quad &= (P^*_{k_2} - \ \phi^*_{k_2}\ C_{k_2})\ D_{k_2}(t) \\ &\quad - (f_{k_2}q_{k_2}(t)^2 + (\eta_{k_2} - \zeta_{k_2})q_{k_2}(t)^2) - C_{0k_2} \end{aligned} \tag{4.23}$$

Different from the instant profit of IaaS provider in the static game in Eq.4.3, we do not consider the rating accumulation in the profit function. The reason is that we already considered the increase of future demand due to the enhanced user rating (quality) in Eq.4.18 and Eq.4.19. The objective function is the total discounted IaaS provider's payoff over the planning horizon $[0 - T]$:

$$\begin{aligned} \text{maximize} \quad & \int_0^T e^{\rho t}\{PP_{k_1}(q_{k_1}(t), D_{k_1}(t))\}dt \\ \text{subject to} \quad & \dot{D}_{k_1}(t) = (\hat{\theta}_{k_1} + \vartheta_{k_1})\ q_{k_1}(t)^{\beta_n} - \check{\theta}_{k_1}\ q_{k_2}(t)^{\beta_n} - \delta_{k_1} D_{k_1}(t) \\ & D_{k_1}(0) = D_{0k_1}, 0 < \beta_n < 1 \end{aligned} \tag{4.24}$$

$$\begin{aligned} \text{maximize} \quad & \int_0^T e^{\rho t}\{PP_{k_2}(q_{k_2}(t), D_{k_2}(t))\}dt \\ \text{subject to} \quad & \dot{D}_{k_2}(t) = \hat{\theta}_{k_2}\ q_{k_2}(t)^{\beta_m} - \check{\theta}_{k_2}\ q_{k_1}(t)^{\beta_m} - \delta_{k_2} D_{k_2}(t) \\ & D_{k_2}(0) = D_{0k_2}, 0 < \beta_m < 1 \end{aligned} \tag{4.25}$$

ρ is a constant discount rate to rebate all the future costs and revenues' streams relative to the present. Note that Eq.4.24 and Eq.4.25 formulate two optimal control problems with the service quality and the cumulative demand as control and state variables, respectively. In the following section we solve these optimal control problems.

4.5.3 Open-Loop Equilibrium Solution

The analysis of differential games relies profoundly on the concepts and techniques of optimal control theory [35]. To study the dynamics of the payoff functions and the paths of control variables, we leverage the Hamiltonian systems. Equilibrium strategies in the open-loop structures can be found by solving a two-point boundary value problem for ordinary differential equations derived from the Pontryagin

maximum principle in Hamiltonian functions. Pontryagin maximum principle gives the necessary condition for a control path to be optimal open-loop control. The optimal control paths of quality are defined as follows.

Definition 4.3. For the IaaS provider k, the quality strategy $q_k^*(t)$ is optimal if the inequality $PP_k(D_k(t), q_k^*(t)) \geq PP_k(D_k(t), q_k(t))$ holds for all feasible control paths $q_k(t) \neq q_k^*(t)$.

To acquire the optimal control, we first formulate the Hamiltonian system of the IaaS providers' payoff which is quite similar to the Lagrangian method that we used in the first game.

$$\begin{aligned} &H_{k_1}(q_{k_1}(t), D_{k_1}(t), \lambda_{k_1}(t), t) = \\ &(P_{k_1}^* - \phi_{k_1}^* C_{k_1})\ D_{k_1}(t) - (f_{k_1} q_{k_1}(t)^2 + \eta_{k_1} q_{k_1}(t)^2) - C_{0k_1} \\ &+ \lambda_{k_1}(t)((\hat{\theta}_{k_1} + \vartheta_{k_1})\ q_{k_1}(t)^{\beta_n} - \check{\theta}_{k_1}\ q_{k_2}(t)^{\beta_n} - \delta_{k_1} D_{k_1}(t)) \end{aligned} \tag{4.26}$$

$$\begin{aligned} &H_{k_2}(q_{k_2}(t), D_{k_2}(t), \lambda_{k_2}(t), t) = \\ &(P_{k_2}^* - \phi_{k_2}^* C_{k_2}) D_{k_2}(t) - (f_{k_2} q_{k_2}(t)^2 + (\eta_{k_2} - \zeta_{k_2}) q_{k_2}(t)^2) \\ &- C_{0k_2} + \lambda_{k_2}(t)(\hat{\theta}_{k_2}\ q_{k_2}(t)^{\beta_m} - \check{\theta}_{k_2}\ q_{k_1}(t)^{\beta_m} - \delta_{k_2} D_{k_2}(t)) \end{aligned} \tag{4.27}$$

The adjoint variable or shadow price (λ_k) associated with a particular constraint is the change in the optimal value of the objective function per unit increase in the right-hand-side value of that constraint, all other problem data remaining unchanged. The economic interpretation of $\lambda_k(t)$ is the value of an additional unit of demand. For given $q_k(t)$, $\lambda_k(t) > 0$ implies that the IaaS provider benefits from current demands. With a zero shadow price $\lambda_k(t) = 0$, the IaaS provider does not take into account the impact of the quality on future user demands. On the other hand, when $\lambda_k(t) < 0$,

the IaaS provider has no motive to sacrifice current profits for future profits, so that it will no longer elevate the service quality.

According to the control theory, the optimal control strategy of the original problem must also maximize the corresponding Hamiltonian function. Thus, based on the Pontryagin maximum principle, all candidate optimal strategies have to satisfy the following necessary conditions:

$$\begin{aligned}\frac{\partial H_{k_1}(t)}{\partial q_{k_1}(t)} = & -2(f_{k_1}q_{k_1}(t)+\eta_{k_1}q_{k_1}(t)) \\ & +\lambda_{k_1}(t)(\hat{\theta}_{k_1}+\vartheta_{k_1})\ \beta_n q_{k_1}(t)^{\beta_n-1}=0\end{aligned} \tag{4.28}$$

$$\begin{aligned}\dot{\lambda}_{k_1}(t) &= \rho\lambda_{k_1}(t)-\frac{\partial H_{k_1}(t)}{\partial D_{k_1}(t)} \\ &=(\rho+\delta_{k_1})\lambda_{k_1}(t)-P^*_{k_1}+\phi^*_{k_1}C_{k_1}, \lambda_{k_1}(T)=0\end{aligned} \tag{4.29}$$

$$\begin{aligned}\frac{\partial H_{k_2}(t)}{\partial q_{k_2}(t)} = & -2(f_{k_2}q_{k_2}(t)+(\eta_{k_2}-\zeta_{k_2})q_{k_2}(t)) \\ & +\lambda_{k_2}(t)\hat{\theta}_{k_2}\ \beta_m q_{k_2}(t)^{\beta_m-1}=0\end{aligned} \tag{4.30}$$

$$\begin{aligned}\dot{\lambda}_{k_2}(t) &= \rho\lambda_{k_2}(t)-\frac{\partial H_{k_2}(t)}{\partial D_{k_2}(t)} \\ &=(\rho+\delta_{k_2})\lambda_{k_2}(t)-P^*_{k_2}+\phi^*_{k_2}C_{k_2}, \lambda_{k_2}(T)=0\end{aligned} \tag{4.31}$$

When only one boundary condition is specified as $D_k(0) = D_{0k}$, the free-end condition is used as $\lambda_{k_1} = \lambda_{k_2} = 0$ at $t = T$. It should be noted that the Pontryagin maximum principle is only a necessary condition, but not essentially sufficient for optimality. Consequently, the solution of the pair quality control in the above equations does not necessarily converge to the Nash equilibrium. To investigate the normality of our defined systems and to assess if Pontryagin can provide a sufficient condition for optimality in our case, we shall derive the monotonicity condition on the adjoint variables in Lemma 1. This condition is important since the adjoint variables

significantly affect the payoff functions in our optimal control-based optimization.

Lemma 4.1. *With positive profit unit margins, we have* $\lambda_{k_1}(t) > 0$ *and* $\lambda_{k_2}(t) > 0$ *for all* $t \in [0, T)$.

Proof. Here we prove the monotonicity of $\lambda_{k_1}(t)$, and the same proof applies for $\lambda_{k_2}(t)$. As stated in Eq.4.29, we have $\lambda_{k_1}(T) = 0$. Therefore, at $t = T$, $\dot{\lambda}_{k_1} = -P^*_k + \phi^*_{k_1} C_{k_1} < 0$, since $P^*_{k_1} > \phi^*_{k_1} C_{k_1}$. So, $\lambda_{k_1}(t) > 0$ as t approaches T. Now, consider $\lambda_{k_1}(t1) < 0$ for any given $t1$. Then we should have $\lambda_{k_1}(t2) = 0$ for some $t2 > t1$ and $\dot{\lambda}_{k_1}(t2) \geq 0$. Consequently, $\dot{\lambda}_{k_1}(t2) = -P^*_{k_1} + \phi^*_{k_1} C_{k_1} < 0$. This is a contradiction and $\lambda_{k_1}(t)$ is proved to be positive during the whole period of time. □

The following proposition is concluded from Lemma 4.1.

Proposition 4.1. *IaaS providers' profit optimization functions have a normal form maximum principle with positive adjoint variables* $(\lambda_{k_1}(t), \lambda_{k_2}(t))$ *associated with* $(q^*_{k_1}(t), q^*_{k_2}(t))$.

Lemma 4.2. *Pontryagin Maximum Principle (in Eq.4.28-4.31) provides the necessary sufficient conditions of optimality and the control path pair of* $(q^*_{k_1}(t), q^*_{k_2}(t))$ *is optimal and unique.*

Proof. Proposition 1 asserts that the formulated profit optimization problems have a normal form. It suffices to prove that the Hamiltonian function is concave in $D_k(t)$ for both providers k_1 and k_2 in any $t \in [0 - T]$. Let $(q^*_{k_1}(t), D^*_{k_1}(t))$ be a pair that satisfies the Pontryagin condition for IaaS provider k_1, with $\lambda_{k_1}(0) = 1$, and for all admissible demand states, the limiting transversality condition holds: $\lim_{t \to T} \lambda_{k_1}(t)(D_{k_1}(t) - D^*_{k_1}(t)) \geq 0$. To prove the concavity of the dynamic function in $D_{k_1}(t)$, the following

condition must hold:

$$\begin{aligned} H_{k_1}(q_{k_1}(t), D_{k_1}(t), \lambda_{k_1}(t), t) - H_{k_1}(q^*_{k_1}(t), D^*_{k_1}(t), \lambda_{k_1}(t), t) \\ \leq \frac{\partial H_{k_1}(t)}{\partial D_{k_1}(t)}(D_{k_1}(t) - D^*_{k_1}(t)) \end{aligned} \tag{4.32}$$

The left-hand-side of the inequality is negative since the Hamiltonian function in the optimal quality path is the maximum IaaS provider profit that is more than its profit at any other path in any time t. Thus, it is enough to prove that the right-hand-side of the inequality is positive. From Eq.4.29, we can see that:

$$\frac{\partial H_{k_1}(t)}{\partial D_{k_1}} = \rho\lambda_{k_1}(t) - \dot{\lambda}_{k_1}(t) \tag{4.33}$$

Replacing Eq.4.33 in Eq.4.32, we get $(\rho\lambda_{k_1}(t) - \dot{\lambda}_{k_1}(t))(D_{k_1}(t) - D^*_{k_1}(t))$ in the right-hand-side. From the transversality condition, we already know that $\lambda_{k_1}(t)(D_{k_1}(t) - D^*_{k_1}(t)) \geq 0$, so it is enough to prove that $\dot{\lambda}_{k_1}(t)$ is negative. It is known in optimal control theory that the motion of shadow price is equal to the negative derivative of Hamiltonian towards the dynamic state, so that $\dot{\lambda}_{k_1}(t) = -P^*_{k_1} + \phi^*_{k_1}C_{k_1} - \delta_{k_1}\lambda_{k_1}(t) \leq 0$. The same logic applies for k_2. □

After proving the monotonicity of adjoint variables in Lemma 4.1 and the sufficiency of the Pontryagin maximum principle in obtaining the optimal solution in Lemma 4.2, we can obtain the optimal control path.

Theorem 4.1. *The finite horizon differential game in Eq.4.24 and Eq.4.25 has a unique Nash equilibrium solution for the two IaaS providers. The optimal quality strategies are given by:*

$$q^*_{k_1}(t) = \left(\frac{(P^*_{k_1} - \phi^*_{k_1}C_{k_1})(\hat{\theta}_{k_1} + \vartheta_{k_1})\beta_n}{2(\rho + \delta_{k_1})(f_{k_1} + \eta_{k_1})}\right)^{\frac{1}{2-\beta_n}}\left(1 - e^{\frac{(\rho+\delta_{k_1})(t-T)}{2-\beta_n}}\right) \tag{4.34}$$

$$q^*_{k_2}(t) = \left(\frac{\left(P^*_{k_2} - \phi^*_{k_2} C_{k_2}\right)\, \hat{\theta}_{k_2} \beta_m}{2(\rho + \delta_{k_2})(f_{k_2} + (\eta_{k_2} - \zeta_{k_2}))}\right)^{\frac{1}{2-\beta_m}} \left(1 - e^{\frac{(\rho+\delta_{k_2})(t-T)}{2-\beta_m}}\right) \tag{4.35}$$

Proof. The two formulated differential equations Eq.4.29 and Eq.4.31 can lead us to the adjoint variables:

$$\lambda_{k_1}(t) = \frac{P^*_{k_1} - \phi^*_{k_1} C_{k_1}}{\rho + \delta_{k_1}} \left(1 - e^{(\rho+\delta_{k_1})(t-T)}\right) \tag{4.36}$$

$$\lambda_{k_2}(t) = \frac{P^*_{k_2} - \phi^*_{k_2} C_{k_2}}{\rho + \delta_{k_2}} \left(1 - e^{(\rho+\delta_{k_2})(t-T)}\right) \tag{4.37}$$

Replacing Eq.4.36 in Eq.4.28 and Eq.4.37 in Eq.4.30 gives us the optimal quality control paths. □

Differential games enable us to analyze the dynamic nature of competition and quality improvement. The following lemmas and propositions are inferred from Theorem 4.1.

Lemma 4.3. *Each provider's quality improvement decreases in its quality development cost.*

Proof. The decrease is straightforward from the first derivative of quality with respect to cost, $\frac{\partial q^*_{k_1}(t)}{\partial(f_{k_1}+\eta_{k_1})} < 0$ and $\frac{\partial q^*_{k_2}(t)}{\partial(f_{k_2}+(\eta_{k_2}-\zeta_{k_2}))} < 0$. However, the cost decrement slope is steeper for big providers due to serving a large number of customers. The difference is specifically reflected in the non-functional costs since functional costs are expected to be alleviated as the service becomes more mature. This corollary is an evidence of the economic benefits of continuous quality improvement for both IaaS providers to have a higher level of quality equilibrium as well as user rating. □

Lemma 4.4. *Higher level of customer loyalty and lower discount factor lead to a higher quality equilibrium for both providers.*

Proof. This corollary simply means the fewer IaaS providers' customer defection rate, the more incentive for the providers to improve their service quality. It can be inferred from the first order conditions for Hamiltonian systems of IaaS provider k_1 (Eq.4.28), where we have:

$$q^*_{k_1}(t) = (\frac{\lambda_{k_1}(t)(\hat{\theta}_{k_1} + \vartheta_{k_1})\beta_n}{2(f_{k_1} + \eta_{k_1})})^{\frac{1}{2-\beta_n}}$$

This implies that the two variables of customer defection rate and discount factor are reflected through the value of the shadow price $\lambda_{k_1}(t)$. As t approaches the end of the time horizon, the negative effect of ρ and δ_{k_1} becomes more evident:

$$\lim_{t \to T} \lambda_{k_1}(t) = \frac{P^*_{k_1} - \phi^*_{k_1} C_{k_1}}{\rho + \delta_{k_1}}$$

□

The same logic is applied for IaaS provider k_2. Thus, as the marginal values of customers drop, the service quality equilibrium shrinks.

Proposition 4.2. *The quality improvement of cloud services is higher in early stages and decreases over time.*

The service quality improvement rate is steeper at the beginning of the time horizon. As t approaches T, the improvement flattens out. The reason can be the maturity of the service, getting maximum user ratings, or adjustment of the service features and support.

Proposition 4.3. *Assuming that 1) both providers make the same revenue per unit service; 2) $\delta_{k_1} = \delta_{k_2}$ with the same user rating sensitivities; and 3) ϑ_{k_1} for IaaS provider k_1 and ζ_{k_2} for IaaS provider k_2 determine the quality level. If smaller providers do not take the advantage of customization and providing value for their*

target segment, then $q^*_{k_2}(t) > q^*_{k_1}(t)$, $\forall t < T$. *The established condition outlines when quality improvement of the bigger providers always dominates the smaller ones.*

In the above propositions and lemmas, we brought a number of managerial insights into attention. We showed that in the early stage, there should be an emphasis on increasing the quality of IaaS. Also, it will be to the IaaS provider's advantage to reduce the quality cost. That will increase the optimum quality level and will give rise to a ripple effect of benefits. The dynamic differential game will be played in a time interval, so that some of the variables may change during that time. In the following section, we will analyze how these variations can affect the optimality conditions.

4.6 Post-Optimality Analysis

The input data in theoretical optimization approaches is not subject to change, however, in real life it might be found impractical. This assumption is rather valid in a static and deterministic environment, while the essence of our problem is dynamic. User demand reflects market behavior that is changing, and in some degree unpredictable. Cost and capacity estimates are sometimes prone to errors and to changes over time due to the dynamic behavior of the market. Therefore, an important question lies in the sensitivity of the obtained optimal solutions to changes in the input parameters.

We investigate the variability of VM request arrival rate due to a future increase in the number of users. Subsequently, this variability may affect the optimal capacity and price as well. As a result, two types of variations may happen in the range of: 1) objective function; and 2) constraints. The objective function's range refers to the range over which capacity and price coefficients can vary, without changing the basis associated with our optimal solution. In this case, for example, by computing the

amount of change in price, we can obtain a new optimal price: $P^*_{new} = P^*_{old} + \Delta P^*$. The constraint's range refers to the user arrival range so that the values of the shadow prices in terms of the defined quality and capacity will remain unchanged.

As the number of users grows, the VM request arrival rate will expand. The value range of l and possible changes to the optimality of the VM price and capacity are investigated in Theorem 4.2.

Theorem 4.2. *IaaS provider best response sensitivity: The optimal solutions obtained for the IaaS provider about price P^* and capacity ϕ^* remain optimum if:*

$$\Delta l < \underbrace{(Q(\phi^* - l) - 1)(\phi^* - l)^2}_{l_{threshold}} \tag{4.38}$$

In that case, the optimal price and capacity vary as follows:

$$\Delta P^* = l\ \Delta l\ C(\frac{\alpha_i + \gamma_i}{\alpha_i + \gamma_i - 1}) \tag{4.39}$$

$$\Delta \phi^* = \Delta l \tag{4.40}$$

Proof. The expressions for the sensitivity derivatives can be derived based on the Kuhn-Tucker conditions. The changes in the optimum values of ϕ^* and P^* necessary to satisfy the Kuhn-Tucker conditions due to a change Δl in the user arrival rate parameter can be estimated as follows:

$$\Delta P^* = \frac{\partial P^*}{\partial l}\Delta l = l\ \Delta l\ C(\frac{\alpha_i + \gamma_i}{\alpha_i + \gamma_i - 1})$$

$$\Delta \phi^* = \frac{\partial \phi^*}{\partial l}\Delta l = \Delta l$$

Earlier, Eq.4.15 proved that $\lambda_k = 0$, and the constraint is inactive in the profit maximization problem in Eq.4.3. Now, Eq.4.14 can be used to determine when an originally inactive constraint becomes active due to the change in VM request arrival rate, Δl. Let us consider the constraint in Eq.4.14 as $g(x) = \frac{1}{\phi^* - l} - Q$. The currently inactive constraint will become critical due to Δl, if the new value of $g(x)$ converts to zero:

$$g(x) + \frac{dg(x)}{dl}\Delta l = g(x) + (\frac{\partial g(x)}{\partial \phi^*}\frac{\partial \phi^*}{\partial l} + \frac{\partial g(x)}{\partial P^*}\frac{\partial P^*}{\partial l})\Delta l = 0$$

Thus, the necessary change to Δl to make $g(x)$ active can be found as:

$$\Delta l = -\frac{g(x)}{\frac{\partial g(x)}{\partial \phi^*}\frac{\partial \phi^*}{\partial l}} = (Q(\phi^* - l) - 1)(\phi^* - l)^2$$

□

The change of the optimal price ΔP obtained from Eq.4.39 shall be examined for its effect on the optimality condition of the user demand size as shown in Theorem 4.3.

Theorem 4.3. *IaaS user best response sensitivity: The optimal solutions obtained for IaaS user i on VM request size x_i^* remain optimum if:*

$$\Delta P^* < \underbrace{\frac{P^*}{\alpha_i + \gamma_i}}_{P^*_{threshold}} \tag{4.41}$$

Proof. To prove Eq.4.41, we should calculate how much λ_{i1} will fluctuate. Similarly, the variation in the value of Lagrange multiplier due to ΔP can be estimated as follows:

$$\Delta \lambda_{i1} = \frac{\partial \lambda_{i1}}{\partial P^*}\Delta P^*$$

The above equation can be used to determine when the originally active constraint

defined for the optimization problem in Eq.4.2 becomes inactive due to the change ΔP. Since the value of λ_{i1} is zero for an inactive constraint, we will have:

$$\lambda_{i1} + \Delta\lambda_{i1} = \lambda_{i1} + \frac{\partial\lambda_{i1}}{\partial P^*}\Delta P^* = 0$$

From Eq.4.9 we calculate λ_{i1} as follows:

$$\lambda_{i1} = \frac{(B_i r_i^{\beta_i}\gamma_i\mu)^{\gamma_i - 1}}{P^{*(\alpha_i+\gamma_i)}}$$

Therefore, the amount of change in the optimal price to diminish its optimality is as follows:

$$\Delta P^* = \frac{-\lambda_{i1}}{\frac{\partial\lambda_{i1}}{\partial P^*}} = \frac{P^*}{\alpha_i + \gamma_i}$$

□

4.7 Experiments and Analysis

As the main purpose of our experiments is to demonstrate the effectiveness of the proposed games, we have to set meaningful data and reasonable game parameters. To do so, we obtained real-world data and previously achieved suitable values for the parameters of the Cobb-Douglas demand function [79]. Initially, we experimented with 300 IaaS users for the small provider k_1 using real customer ratings to investigate the sensitivity of pricing formula to VM request arrival rate. The data was collected from the Trust Feedback Dataset, provided by Noor et al. [64] in the CloudArmor project[5].

To simulate the differential game, we assumed VM request arrival rate to be 300 tasks per hour for k_1 and 900 tasks per hour for k_2, that are realistic and commonly

[5] `https://cs.adelaide.edu.au/~cloudarmor/ds.html`

used values for cloud services [41]. The price for k_1 is borrowed from a local IaaS provider in Malaysia, called exabytes[6]. Due to the earlier explained reasons, k_2 price will be less or at most the same. Both cases are to be considered in our experiments. To obtain the VM process rate (capacity), which is the minimum speed of 2mbps, we referred to the IaaS promised QoS in the SLA statements. Since there is no information available about the providers' cost, we approximate the cost of k_1 to the cost of renting large cloud infrastructure from Google (per VM per hour) and the cost of k_2 is set to be 100 times less.

The value of ζ_{k_2} (the discounted non-functional cost) is approximated using the Eta-Squared statistics of the ANOVA analysis on the acquired user ratings given to non-functional quality features. The reason behind using Eta-Squared comes from the fact that the average Eta-Squared of some non-functional features (e.g., availability and response time) reflects the importance of these parameters on customers' demand. In fact, we used Eta-squared to measure the effect size of the independent variables (the non-functional attributes). On the other hand, there is no feature representing the personalization value to customers to be used for ϑ. So, we run our experiments by giving different values to ϑ. The rest of the parameters are assigned based on the past literature [5, 24, 58]. Table 4.3 depicts the utilized values of variables in the experiments. The time axis is normalized to the (0-1) interval.

As discussed previously, there is no similar work to our model or related experiments to be compared to. For this reason, only the results of our model are reported.

[6] `www.exabytes.my`

Table 4.3: Assigned variables' values in simulation

Variables	k_1	k_2
P	0.18	0.18, 0.13
C	0.000076	0.00000076
δ	0.001	0.001
ρ	0.005	0.005
$\hat{\theta} = \check{\theta}$	0.3	0.3
η	0.5	0.5
f	0.5	0.5
ζ_{k_2}	NA	0.7
ϑ	0, 0.7, 0.9	NA
l	300	900

4.7.1 Significance of Quality over User Rating

A one-way ANOVA was conducted to assess the effect of the provided quality on the user rating. The ANOVA test was performed over 2000 user ratings given to 78 distinct cloud services considering 8 attributes representing functional and non-functional quality features. Given that the significance value (p) is less than the α-value ($\alpha = .001$) for all quality features, as reported in Table 4.4, we can rest Assumption 1 and claim that quality attributes are strongly positively correlated with the overall user rating score.

The analysis of variance and Eta-squared values showed that the effect of the technical support attribute was the most significant criterion. It was followed by customer service, response time and availability. Taken together, these results suggest that high levels of more tangible and measurable qualities have more effect on the user rating score.

The ANOVA results proved that user rating has a strong tie with after-sales service, and customer support has turned into a crucial tool in an organization's arsenal of sales tools. In classical business models, there is little incentive to provide excellent customer support since the majority of the revenue from a customer is

Table 4.4: ANOVA Test results

Service attributes	Sum of Squares	df	Mean Square	F	Sig. (p)	Eta-Squared
Technical support	3215.603	1907	525.621	1701.668	.000	**.817**
Response time	768.870	698	145.287	537.118	.000	**.756**
Availability	2212.676	1414	331.846	844.832	.000	**.750**
Speed	1472.511	742	218.983	427.415	.000	**.744**
Ease of use	1541.725	1289	283.299	891.095	.000	**.735**
Accessibility	533.575	630	97.636	427.327	.000	**.732**
Operation& management features	853.372	617	149.497	358.836	.000	**.701**
Storage space	846.267	601	115.788	180.430	.000	**.547**

already secured. In today's subscription model, however, the equation is almost reversed. Once a service is sold, the IaaS provider receives a very small fraction of the lifetime revenue at the beginning of the transaction. Afterwords, the support team is under a great pressure to keep the customer satisfied. This satisfaction is also crucial to enhance the customer loyalty that has a significant impact on quality equilibrium as asserted by Lemma 4.4.

4.7.2 Sensitivity Analysis

In order to evaluate the VM request arrival rate threshold and optimality of price, we simulate an increasing number of VM request arrival with a fixed price for IaaS provider k_1. Given the speed of 2mbps, Q (for k_1) is 900 in an hour. According to our obtained formula in Eq.4.38, we have: $\Delta l < 900(0.5)(1.5)^2$, that makes a critical value of change to VM request arrival at about $\Delta l < 1000$. This means that if this provider experiences an increase of 1000 VM request arrivals, it needs to recalculate its pricing strategy since it is not making an optimized profit. This sensitivity is illustrated in Figure 4.3. Once the VM request arrival rate crosses the threshold

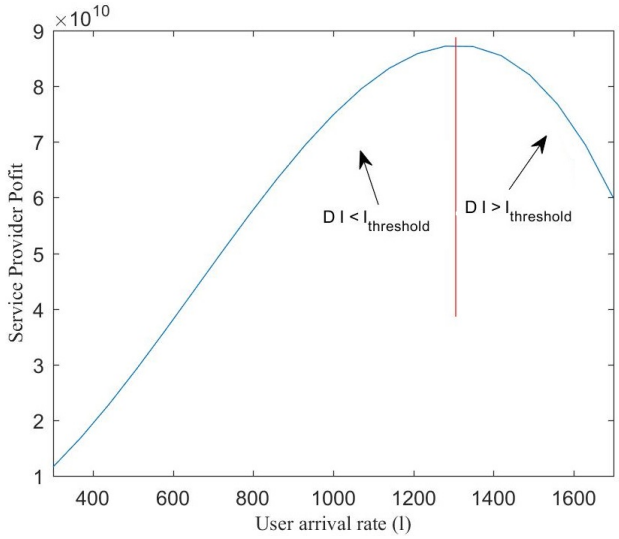


Figure 4.3: Sensitivity of pricing optimality to the increase of VM request arrival rate

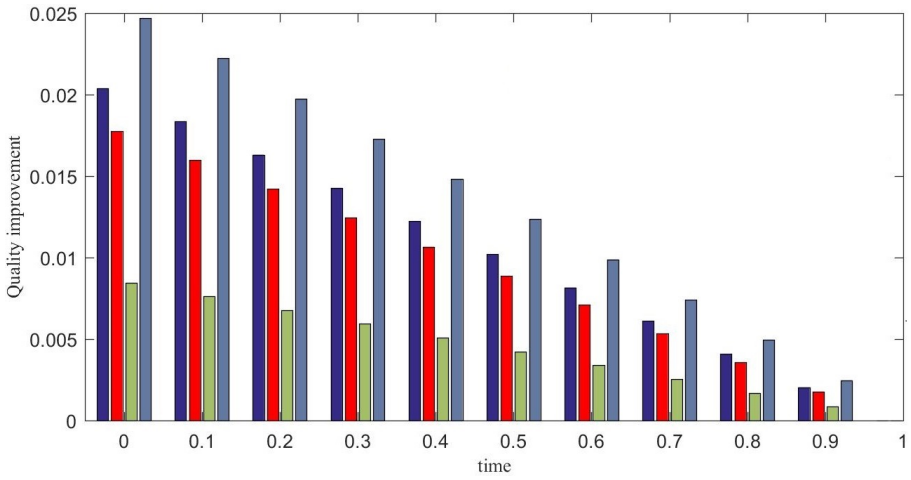
($\Delta l = 1300 - 300 = 1000$), the profit starts sinking.

4.7.3 Quality Improvement Impact on User Demand and Profit

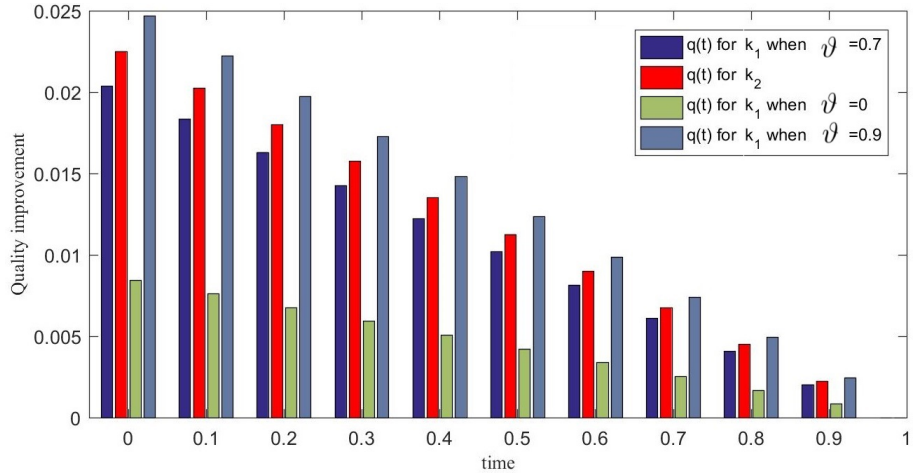
During the first game, the IaaS provider has to set the optimal price based on the predicted user demand response given the offered price. The best pricing strategy should consider users' reactions towards the given price. Nevertheless, there are two possible cases for IaaS k_1's price to be considered: 1) $P_{k_1} > P_{k_2}$; and 2) $P_{k_1} = P_{k_2}$. It was learned that the smaller provider faces higher costs, so that it cannot offer a price cheaper than the large provider. Thus, k_1 can offer either the same price as k_2 or a higher price. The following sections provide more details on these two scenarios.

Scenario 1: higher pricing

To assess the role of the competitive advantage of k_1, we assign three different values to ϑ_{k_1}, each representing a specific market positioning:



a) when $P_{k_1} > P_{k_2}$



b) when $P_{k_1} = P_{k_2}$

Figure 4.4: The IaaS quality improvement of k_1 and k_2

1. $\vartheta_{k_1} = 0$: in this case, the IaaS provider is not using its strategic advantage and does not target any market niches nor offer customization.

2. $\vartheta_{k_1} = 0.7$: this means the IaaS provider is providing added value for customers as worthy as the non-functional quality advantages offered by the existing competitors.

3. $\vartheta_{k_1} = 0.9$: this case illustrates a situation where the IaaS provider is making extra effort to provide more value to the customers than the existing offers.

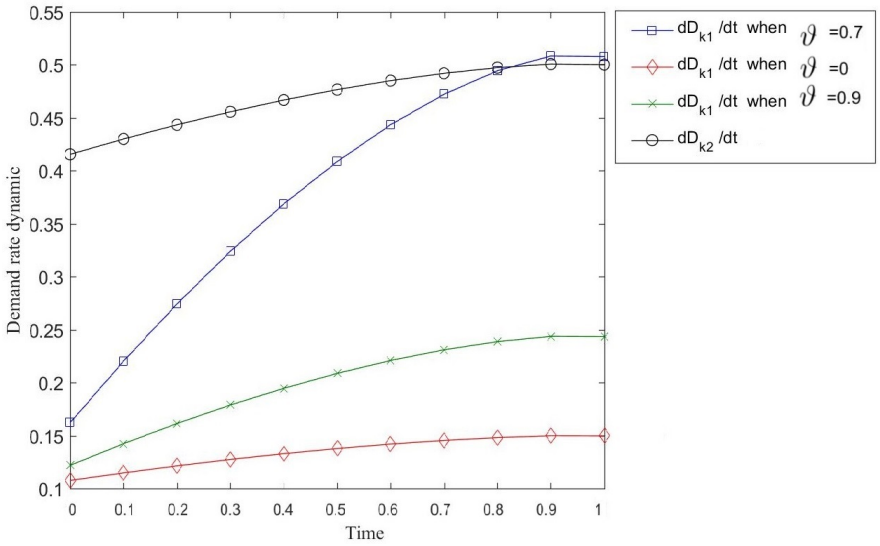


Figure 4.5: The change of user demand when $P_{k_1} > P_{k_2}$

Figure 4.4 (a) presents the quality improvement steps over time for k_1 and k_2 when $P_{k_1} > P_{k_2}$. Agreeing with Proposition 4.2, the quality improvement rate is steeper at the beginning and flattens out at final stages. When k_1 is providing the same value as k_2, it needs to put more effort on quality than k_2. The difference of this extra quality is higher in the first steps, but reduces over time. Thus, k_1 can, for instance, expand the infrastructure to preserve a more capacity and networking bandwidth, or reorganize the tenant clusters. Customer support is specifically important where the small provider operates to provide a more customized solution. Examples of less costly quality improvement would be offering creative features and highly customized and targeted services.

Mapping the defined quality improvement to user rating increment gives the opportunity to identify how the users feel about the trade-off between the price and quality. When the users are paying higher price and receiving lower quality, they would expect to see much higher added-value to their businesses. The strategic benefits will satisfy their expectations and would rise their ratings. This is highly significant for smaller providers to plan the right amount of investment to improve the quality in

their early stages of development. Clearly, the case where k_1 is making extra effort on providing customer value requires additional improvement.

The effect of such a quality (rating) improvement on the user demand rate is quite interesting. As shown in Figure 4.5, k_1 gains the highest change in user demand when it is providing the same value as k_2. The change it experiences exceeds that of k_2. This can be because of receiving a closer rating to k_2, which can reflect how valuable is providing such targeted services to customers. It is not surprising that the demand rate increases very little when the IaaS provider is not offering any special value. Remarkably, providing extra value does not necessarily lead to a higher demand rate. Overdoing that may result in limiting the range of the targeted customers. If the IaaS provider offers very specific and customized services, it may narrow its range of clients and miss the market share that it could obtain. Therefore, finding a suitable strategy is essential for the IaaS provider that wants to compete and earn its share of a profitable but competitive market.

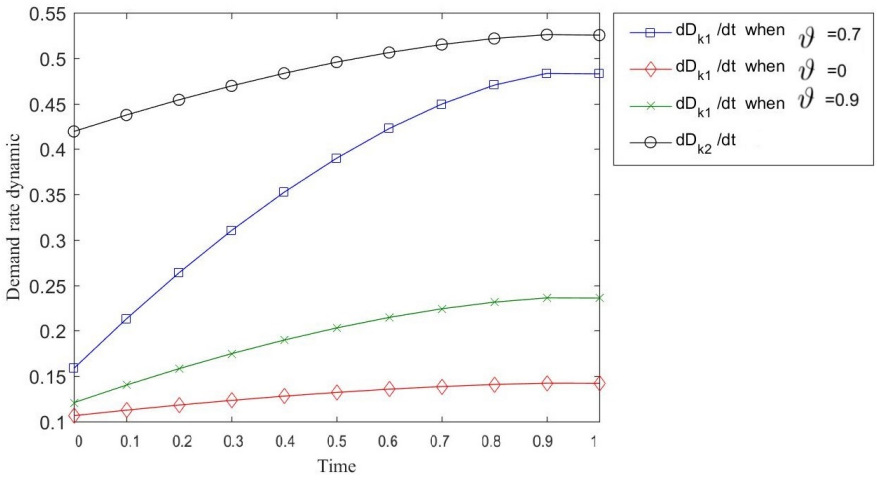


Figure 4.6: The change of user demand when $P_{k_1} = P_{k_2}$

Scenario 2: equal pricing

In this scenario, the small IaaS provider sets the same price as the existing IaaS in the market. Unlike the first scenario, this time k_1 has less improvement of quality than k_2 when $\vartheta_{k_1} = 0.7$ as shown in Figure 4.4 (b). This is due to the fact that k_1 is lowering its pricing down to the same amount as k_2, while its cost is higher. Besides, users of k_1 have already gained the benefit of having lower prices, so the provider does not have to offer higher quality to cover the benefit of price and gain higher ratings. Meanwhile, as expected, no difference is observed between the two scenarios in the amount of the quality improvement when ϑ_{k_1} is 0 or 0.9. This is reasonable because providing zero value (resp. a very high value) demands the same quality improvement regardless of the pricing strategy.

Figure 4.6 illustrates the variation in demand rate over time. When k_1 sets equal pricing as k_2, its demand increase does not reach the change of k_2's demand, which remains higher. Unexpectedly, the equal pricing strategy mainly affects k_2's demand rather than k_1. This event can be related to the impact of quality improvement on the user demand rate. When k_1 sets higher pricing, it can afford more improvement leading to enhanced rating, and its users demand gets slightly higher. Meanwhile, k_2 takes the most advantage of k_1's lower rating improvement to attract more users. It can be inferred that customers prioritize quality over price, which confirms the movement of Cloud 2.0. The trend of k_1's user demand rate variation offering the highest and lowest values does not present any significant change.

Provider's profit and users' loyalty

Total variations of IaaS provider's profit with both pricing strategies are presented in Figure 4.7. Since the trend of profit for k_1 when $P_{k_1} = P_{k_2}$ was almost the same as when $P_{k_1} > P_{k_2}$, we provided only one plot for each different case of ϑ_{k_1}. However, as

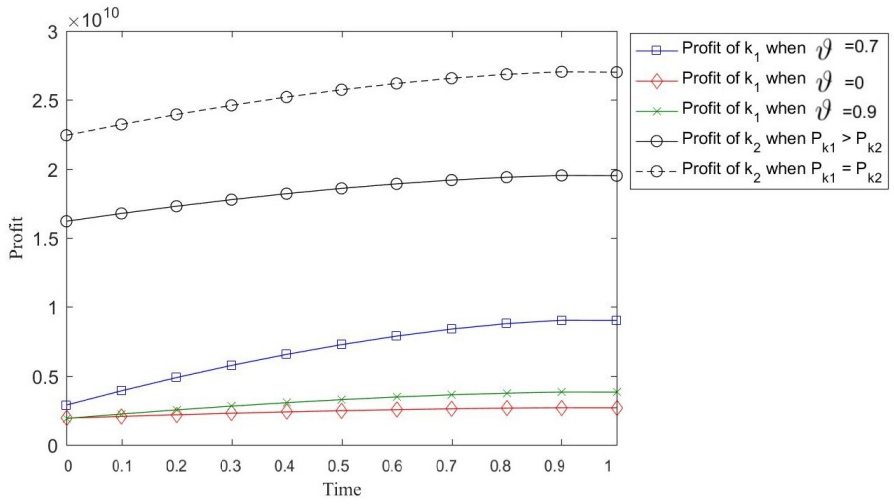


Figure 4.7: IaaS provider's profit taking different quality controls and pricing strategies

the profit of k_2 differs significantly depending on the pricing strategy ($P_{k_1} = P_{k_2}$ or $P_{k_1} > P_{k_2}$), two separate plots are depicted. This figure provides a very useful insight for small IaaS providers by showing that pricing does not alter profit optimization as long as they provide a quality level adjusted with that pricing level. Higher price demands more quality improvement to meet the user expectation and gain high rating. Consequently, the IaaS provider undergoes the burden of the cost associated with that improvement such as increasing the number of servers to reserve more capacities. Here, k_1 can rely on its identified market segment behavior through obtaining its sensitivity to price and rating to decide about setting a reasonable price.

Although the pricing strategy does not significantly affect the profit of k_1, it has enormous influence on the profit of k_2. The reason is that k_2 has already established its reputation as an IaaS leader and obtained a high rating, so that its quality improvement is saturated. In fact, when k_1 fails to attain the customers looking for high quality and customized services, k_2 attracts them. However, k_1 can recompense its profit with higher service price. In this case, k_1's customers will mainly constitute the new public IaaS adopters who could not enjoy the cloud computing

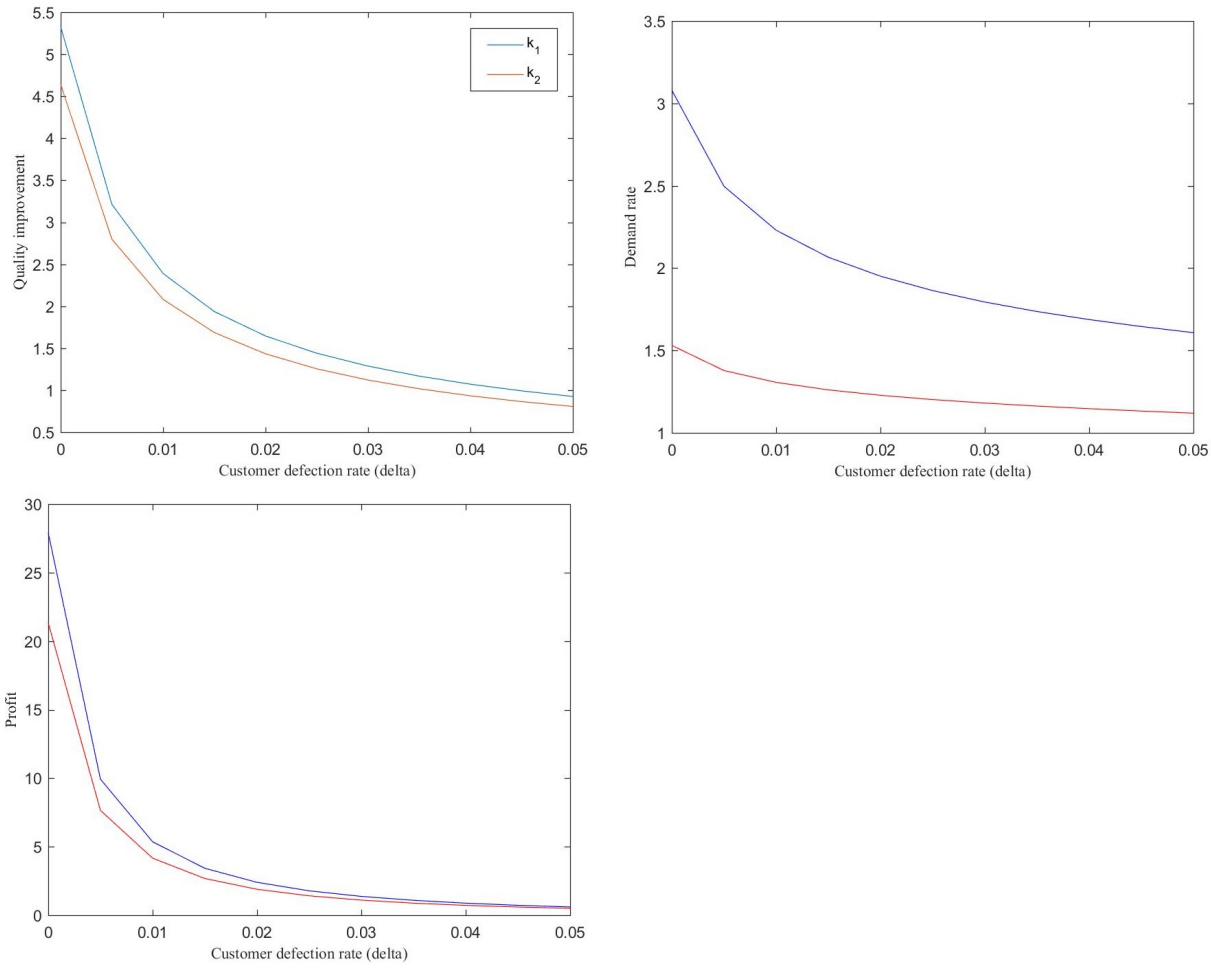


Figure 4.8: Customer loyalty effect on quality and profit

benefits due to their regional, national or international barriers or because of some very customized needs. Although customization is essential for some businesses, yet the quality features of the services are very important. The majority of the users are not willing to sacrifice one for the other. Thus, if the smaller IaaS providers are ready to compete and gain more market share through their own market segment, they need to supply a reasonable amount of IaaS non-functional quality.

We further investigate the importance of customer loyalty on IaaS providers' revenue. Being loyal to an IaaS provider in today's subscription model has mutual benefit for both, the users and providers. As Figure 4.8 illustrates, customer loyalty

is a key motivation for quality improvement for both providers, k_1 and k_2. The effect of customer loyalty is intense when the customers exhibit a highly loyal behavior. In this case, IaaS providers commit themselves to provide high quality services. However, customers with low and even medium loyalty do not make a significant difference. Consequently, as the customer defection rate increases, the user demand and provider's profit drop. The new and small providers are slightly more vulnerable to customer defection, in particular when it comes to future demand provisioning. In fact, the new IaaS providers need to establish their credibility by increasing their users satisfaction and attracting high users' ratings. They also have a limited range of customers compared to the established providers. However, it is most likely that customers who receive customized and targeted infrastructure services would be more loyal to their providers since it is unlikely to find such services anywhere else.

In summary, IaaS quality, specially customer support, has a strong correlation with users' ratings. It offers a managerial point for IaaS providers to increase the customer loyalty through not only customized services, but also fully commitment in after-sale support. The results show that when the IaaS users are paying higher and receiving lower non-functional qualities, they would expect to see much higher added-value from the IaaS provider. The strategic benefit satisfies the user's expectation and rise the provider rating. The small IaaS providers gain the highest change in user demand when they provide the same quality as the established ones. Notably, providing extra value does not necessarily lead to a higher demand rate as it might limit the range of the targeted customers. Improving the quality and ratings of a small and new IaaS provider specifically increases its demand rate and profit in the both pricing scenarios, higher and equal. Setting equal pricing for both IaaS providers mainly favors the established provider. When the smaller provider sets equal pricing, it cannot afford more improvement that leads to a lower user satisfaction. This failure

to attract high user ratings makes a perfect situation for its competitor to attract them, although it does not harm its own profit since it gains the difference of the demand with the difference in the price and quality, and maintains the customers who have no choice but their customized services.

4.8 Conclusion

This chapter tackled the issue of oligopoly IaaS market that neglects the user satisfaction and threatens the grows of the cloud market industry. A conceptual game theoretical framework with two games, namely Stackelberg and differential has been introduced and designed to allow new and even small IaaS providers to obtain a market share using their own strategic advantages. The theoretically obtained results were confirmed by experiments using real-world dataset. It was found that the user demand from small IaaS providers increases the most when these providers provide added-value services equal to the value offered by the existing providers. Regardless of the pricing strategy (higher or equal), improving the quality and ratings of a small and new IaaS provider increases its demands' rate and profit. However, the best strategy for small IaaS providers is to set higher price and improve the quality of their provided added-value solutions, specifically in the early stages of development. The reason is that service customization increases the customer loyalty in today's subscription cloud economy model, where customers are free to defect anytime. Higher customer loyalty elevates the provider's profit and increases the quality equilibrium. Higher level of service quality leads to a higher user satisfaction and improves the small IaaS provider's market position through higher ratings. This research drew many technical, strategic and managerial insights to guide IaaS providers on how to utilize their strengths in deciding on future opportunities and target markets. By offering

value beyond simply providing computing resources, the IaaS provider will play a strategic role in the future of Cloud 2.0.

Chapter 5

Cloudchain: A Blockchain-Based Coopetition Differential Game Model for Cloud Computing

In this chapter, we introduce, design and develop *Cloudchain*, a blockchain-based cloud federation, to enable cloud service providers to trade their computing resources through smart contracts. Traditional cloud federations have strict challenges that might hinder the members' motivation to participate in, such as forming stable coalitions with long-term commitments, participants' trustworthiness, shared revenue, and security of the managed data and services. Cloudchain provides a fully distributed

structure over the public Ethereum network to overcome these issues. Three types of contracts are defined where cloud providers can register themselves, create a profile and list of their transactions, and initiate a request for a service. We further design a dynamic differential game among the Cloudchain members, with roles of cloud service requesters and suppliers, to maximize their profit. Within this paradigm, providers engage in coopetitions (i.e., cooperative competitions) with each other while their service demand is dynamically changing based on two variables of gas price and reputation value. We implemented Cloudchain and simulated the differential game using Solidity and Web3.js for five cloud providers during 100 days. The results showed that cloud providers who request services achieve higher profitability through Cloudchain compared to those providers that supply these requests. Meanwhile, spending high gas price is not economically appealing for cloud requesters with a high number of requests, and fairly cheaper prices might cause some delays in their transactions during the network peak times. The best strategy for cloud suppliers was found to be gradually increasing their reputation, especially when the requesters' demand is not significantly impacted by the reputation value. This chapter is published in [82].

5.1 Introduction

To mitigate the issue of underutilized and over provisioned computing resources, cloud providers scaled their pool of resources by forming cloud federations to maximize their profit and provide guaranteed QoS [14, 32, 47]. In spite of the prominent federation advantages, cloud providers are reluctant to participate in due to some strict challenges, mainly: 1- The stability of a federation is a key factor for the cloud providers to ensure their profitability [32]. Such a stability requires long-term

commitments from the providers, which is very hard to obtain. 2- A federation needs to address the complications of a fair revenue sharing model to warrant that each cloud provider will gain a revenue according to the amount of computational resources contributed to the federation. 3- The presence of unknown and untrusted participants in a federation can degrade the QoS of the federated services [67]. The trust issue limits conventional federations to enroll only trusted providers and disregard the new ones. 4- Having a large pool of computing resources in a grand coalition might increase the opportunity of botnet attacks. Meanwhile, forming a small federation might hinder the revenue maximization of its participants [4]. 5- There are some security and privacy concerns regarding the managed data and services as well as the creation and management of the cloud federation itself. All the necessary information to manage a federation is usually maintained in a centralized trusted third party. This implies that a federation must maintain roles concerning the authorization to manage the participants' information that yields, which makes not only a single point of failure, but also raises trustfulness concerns [47].

Contributions: This research overcomes the traditional cloud federation issues by contributing a novel architecture and an innovative strategic game model:

1. To provide a practical cooperative solution that any cloud provider can embrace regardless of their market position and trustworthiness, we advocate a fully distributed architecture with a democratic governance structure, called *Cloudchain*. To effectively enforce such a structure, Cloudchain proposes an innovative exploitation of blockchain to prompt and support interoperability and coopetition among the cloud providers over the public Ethereum network. Within Cloudchain, cloud providers endeavor to overcome the resource limitation in their local infrastructure by outsourcing their customers' requests to other members of the Cloudchain. Moreover, it allows providers to access

underutilized resources and lease them at cheaper prices. By leveraging blockchain-enabled smart contracts [78], we eliminate the need for trust in the federation and reduce barriers of entry [43].

2. To incentivize the cloud providers and help them make wise decisions about the utilization of Cloudchain, a dynamic differential game is designed, solved and simulated. This game aims to maximize the profit of the Cloudchain members who cooperatively compete while their service demand is dynamically changing. Two variables are considered to impact the cloud provider's revenue, the demand variability and the quality of the provided service: gas cost and reputation value. Gas is a proportional amount that Ethereum pays to motivate the miners to participate in the mining process and to supply a fair compensation for their computation effort [51]. Reputation value is defined to assign a credibility proportional to the quality that a Cloudchain member provides.

We implement the Cloudchain prototype using Solidity and Web3.js which is available open source in Github[1]. We further simulated the differential game using the Gratner's rating dataset[2] where five real-world providers trade their services. Despite being costlier to transact for cloud-service providers who request a service rather than supply, the obtained results proved it is economically justified to adopt Cloudchain.

5.2 Related Work

The literature about cloud-providers cooperation focuses on federation formation as coalitional games where capacity and revenue are shared [63]. Coronado et al. had an intensive investigation on federation-formation variables among cloud providers,

[1]https://github.com/kavehbc/Cloudchain
[2]https://www.gartner.com/reviews/market/public-cloud-iaas

including revenue sharing mechanisms, capacity and cost disparity, and the presence of a big competitor [17]. They defined revenue sharing mechanisms as the most important factor. Among these mechanisms, shapely value and outsourcing models had the least and best performance, respectively. They indicated that collaborating cloud providers can implement a mechanism in which a provider outsources some of its business and gets a percentage of the revenue. The outsourcing model allows the provider to keep some of the revenue of its secured business, even though it is not able to fulfill that business alone. The authors had an insight through the demand peaks and concluded that cloud providers tend to stay in outsourcing collaboration when the demand is high. However, interoperability, trust among cloud providers and service quality or SLA are not considered in their study. The findings from this study confirm the superiority of outsourcing in terms of maximizing the profit of cloud providers, which is what we are proposing in this paper in addition of having the advantage of coopetition among cloud providers. The fact that providers tend to collaborate when they face a hike in their demand, reinforces the consideration of a dynamic and long/short-term federation like Cloudchain. The challenges of interoperability and trust issues among cloud providers are also addressed by the blockchain platform we propose in this paper. Another cloud outsourcing model has been performed by Chen et al. [14] who analyzed the interrelated workload factoring and coalition formation game among private clouds. The authors integrated two types of federations: 1) vertical (outsource workload to public clouds), ad 2) horizontal (share resources with other private clouds. Their experiments found this approach to be promising to improve the cloud's service quality and decrease the delay by 11%. However, their research was limited to service quality and economic aspects of stable cooperation patterns without considering other challenges of a traditional federation explained in the previous section.

Very few efforts have been made to study the potential of blockchain in real-world applications despite its great potential for businesses to share data and collaborate in a secure and customized manner [53]. According to Tractica, a market research firm, the annual revenue for enterprise applications of blockchain is estimated to increase to $19.9 billion by 2025 [39]. The majority of studies about blockchain's application have focused on finance [85], energy [60] and IoT applications [102]. In cloud computing and service industry, to the best of our knowledge, there has been only one academic initiative that proposed a cloud marketplace based on the blockchain technology. Klems et al. designed Desmaa, a conceptual framework for trustless intermediation in service marketplaces using blockchain [43]. This conceptual framework modeled the interactions between a service provider and a service consumer and tried to overcome problems of conventional marketplace systems, such as barriers of entry and transaction costs. Yet, the outsourcing model with collaboration and competition among cloud providers themselves are not considered in their research. Moreover, the providers' profit and the best strategies for utilizing this marketplace is not elaborated nor modeled. Even though the authors developed a prototype, no evaluation and validation against real-world's scenarios were provided.

5.3 Cloudchain Architecture

Cloudchain incorporates three types of smart contracts including a set of executable functions and state variables. Similar contracts are proposed in [3] in the context of medical data management. *Contract 1* (*C1*) or Cloudchain Registery (CCR) is a global contract that maps cloud providers identification values (including *Name*, *Reputation Value*, *Computing Capacity* and *Storage Capacity*) to their Ethereum address identities (equivalent to public keys). The reputation values can be computed

from the customers' ratings given to each provider through online rating platforms. Policies coded into the contract can regulate registering new providers or changing the mapping of the existing ones. The cloud provider registration can be restricted only to certified providers. CCR also maps identities to the Cloudchain Contract (CCC) address on the blockchain, where a special contract regarding each provider profile and list of services is recorded.

Contract 2 (*C2*) denotes Cloudchain Profile (CCP). It holds a list of references to CCC, representing all the participants' previous and current engagements with other nodes in the system. CCP also implements a functionality to enable provider notifications. Providers should register their requests in this contract. Each transaction list stores a status variable. This indicates whether the transaction is newly established, awaiting pending updates and has or has not been completed. This contract is important as it stores the address of all new CCC contracts, without which Cloudchain can simply lose the track of all the contracts.

Contract 3 (*C3*) represents the Cloudchain Contract (CCC). It is issued between two nodes in the system when one node accepts and provides the requested service for the other. The beneficiaries can also complete, or cancel the contract. Once the contract is completed or canceled, the contract balance would be transferred to the supplier-, or requester address respectively, and the contract status would also be updated. There are two approaches to reduce the size of the data as well as the cost of transactions over Cloudchain. The first approach is a common practice for data storage in smart contracts and consists of storing raw data off-chain, and meta-data, small critical data, and hashes of the raw data on-chain [94]. However, the selection of off-chain data storage has some concerns regarding the interaction between the blockchain and the off-chain data storage. The other approach is to provide a common glossary among cloud providers to define the generic terms and policies to be referred

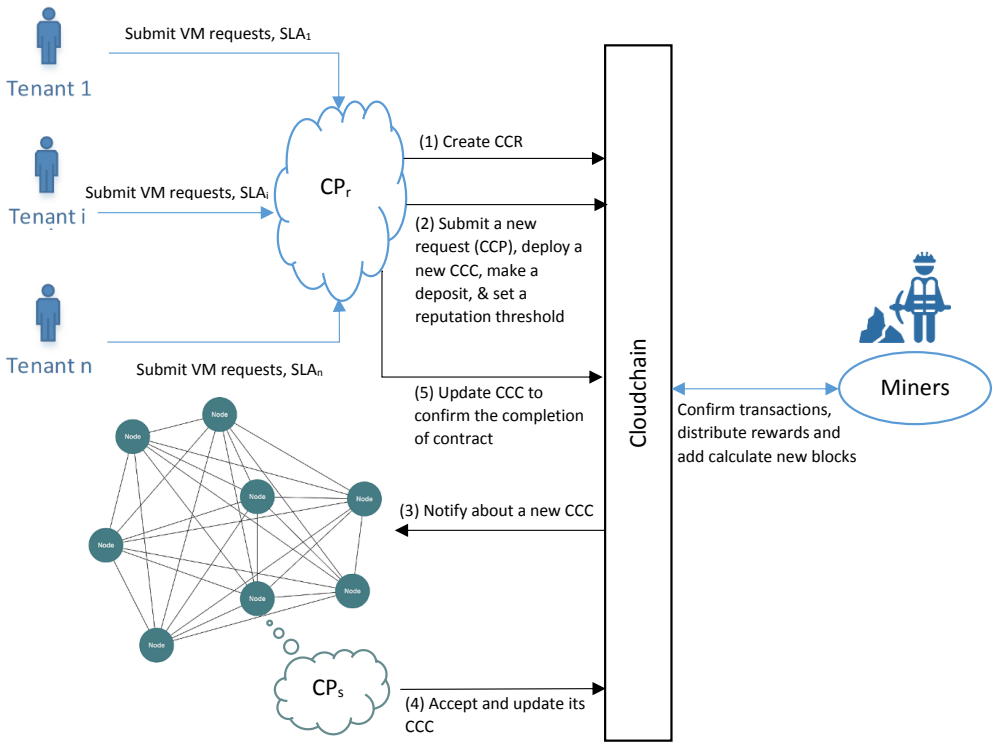


Figure 5.1: Cloudchain interactions

to in the contract.

Fig. 5.1 provides the steps taken by the Cloudchain members to register and establish their requirements by interacting via Cloudchain. In step 1, a provider registers in CCR. Each registered user is assigned with a public key pair. Guaranteed SLA tenants require performance consistency and scale predictability. When a member faces a computing-resources deficiency to meet its end users' demand with guaranteed SLA, it can submit a request for a service using CCP to deploy a CCC to the blockchain in step 2. Requesters are required to pay a deposit in advance and it is stored in the contract. Meanwhile, a rule for providers is set by the requesters to ensure that qualified providers could ultimately receive the task, e.g. reputation value threshold. Function calls on contracts are transactions, and those which update the contract storage need to be validated by miners. Once a new block is mined with the

newly linked CCC, it would be broadcasted to other nodes in step 3. Through step 4, the first node that accepts the request should update the respective CCC contract. Each provider who accepts a task should deposit some coins or its reputation value to guarantee the quality of the task. The contract termination and delivery of the requested service have to be confirmed by the service requester in step 5. The requester is required to rate the supplier based on the received service quality.

5.4 Cloudchain Members' Revenue Optimization

A true blockchain-led federation will not happen unless cloud providers are widely engaged and able to manage the costs and properly play their role. The use of a differential game is motivated by the need to model the time constrained and dynamic strategies of selfish cloud providers willing to maximize their own revenue. Let us consider two typical cloud providers (CP) over Cloudchain, CP_r as the provider who is facing a peak time and is going to request some VMs from other Cloudchain members, and CP_s as the Cloudchain member that has some idle servers and is willing to rent them out with the price offered by CP_r. For simplicity and without losing generality, we will focus on a single VM type, with ϕ denoting the capacity and the process rate of VM instances that can be hosted by a typical CP_j that can be CP_r or CP_s. To make a request and create a contract, CP_r has to define the price of gas G_r for the created transaction (e.g. 5 gwei). If the price is high enough, the transaction will be executed sooner, since miners will execute transactions with the highest gas price first. If the price is set too low, CP_r may end up waiting longer for execution of its transaction and distribution of its request. This waiting time may degrade the service quality for its users and hinder its profit. On the other hand, setting a high gas price for every single transaction and update incurs higher costs. So, the gas price is a

decisive factor in profit optimization and we define it as the control path at time t for CP_r, denoted by $G_r(t)$. In this game, VM price is assumed to be given by CP_r.

To be qualified to supply a cloud service, the provider CP_s has to maintain a good reputation R_s which is given based on the quality of service for end users and the quality of collaboration (e.g. speedy communication) with the cloud provider that requested the service, CP_r. Even though the reputation value is given by CP_r and not CP_s itself, yet it has a control over this value through the service quality and gas price of its own transactions. Therefore, $R_s(t)$ is considered as the control path of CP_s to coopete with CP_r within Cloudchain to gain higher profit. Table 5.1 provides a summary of the notations used in our model.

In order to capture the demand elasticities and variations specific for each user, we define the user demand using the Cobb-Douglas function that models well these elasticity aspects in terms of price and reputation, adopted from [79]. It is assumed that the user will have the opportunity to check the cloud provider rating that represents the actual user satisfaction level and reputation value defined through Cloudchain. The user demand function is defined as follows:

$$D_u = \mu \; p^{-\alpha_u} \; R^{\beta_u} \tag{5.1}$$

In the mining race, miners have to compete to solve proof of work and propagate the block to reach consensus. The new blocks' generation follows a Poisson process with a constant rate $\frac{1}{\Gamma}$ throughout the whole Cloudchain network [45]. Before the race, miners collect their selected pending transactions into their blocks with a total gas amount of $\sum_{j=1}^{k} M_j$. When miner j propagates its block to Cloudchain for consensus, the time for verifying each transaction is affected by the size of transactions M_j. The first miner j who successfully has its block achieves consensus will be rewarded based

Table 5.1: Notations used in Cloudchain

u	End user index
$r, s \in \{1, 2, ..., j, ..., k\}$	Requester and supplier in the set of k cloud providers in Cloudchain
G	Cost of gas
M	The amount of the cumulated gas for each block
$M^{'}$	The amount of required gas for each transaction
X	Number of the transactions occurring over Cloudchain
ω	Rate of transactions arrival for a CP in $[0 - T]$
Γ	Rate of the block generation for miners in Cloudchain
$\phi/\phi^{'}/\phi^{''}$	Provider's active/idle/mining capacity
$p/p^{'}$	Price per VM for the end user/for the members of Cloudchain
R	Reputation value of cloud provider in Cloudchain
Rw	Reward value of mining
τ	Block propagation time
η	Rate of the impact of M over τ
$\check{\theta}$	Rate of CP_r demands rise due to the higher quality services of CP_s
$\hat{\theta}$	Rate of $D_r^{'}$ demands increase due to higher reputation of CP_s
ψ	Rate of CP_r demands increase due to higher gas and higher quality
α_u/β_u	CP price/rating variation for user u
δ	Demand decay rate
μ	The amount of VMs
$C/C^{'}$	Cost of the primary/outsourced capacity

on the amount of the assigned capacity $\phi_j^{''}$. Thus, miner j's expected reward $Rw_j(\phi_j^{''})$ is:

$$Rw_j(\phi_j^{''}) = Rw_j \mathbb{P}_j(\phi_j^{''}, M_j) \tag{5.2}$$

where $\mathbb{P}_j(\phi_j^{''}, M_j)$ is the probability that miner j receives the reward by contributing a block. To win the reward, provider must perform a successful mining and instant propagation. The miner may fail to obtain the reward if its new block does not achieve consensus as the first. This kind of mined block that cannot be added to the blockchain

is called orphaned block. The block containing a larger size of transactions has a higher chance of becoming orphaned since a larger block requires more propagation time, thus, causing a higher delay for consensus. As the arrival of new blocks follows a poisson distribution, miner j's orphaning probability, $\mathbb{P}_j^0$, can be approximated as:

$$\mathbb{P}_j^0 = 1 - exp(-\frac{1}{\Gamma})\tau_j \tag{5.3}$$

Here, we assume miner j's block propagation time τ_j is linear with the size of transactions in its block, $\tau_j = M_j\eta_j$, where η_j is a constant that reflects the impact of M_j over τ_j. Therefore, we obtain the reward probability as follows:

$$\mathbb{P}_j(\phi_j^{''}, M_j) = 1 - \mathbb{P}_j^0 = \phi_j^{''} e^{-\frac{1}{\Gamma} M_j \eta_j} \tag{5.4}$$

Substituting Eq.5.4 into Eq. 5.2 provides an estimation of total revenues that CP_j may obtain by attending the mining tournament. To model the transactions' distribution, we use the compound Poisson process, which is a generalization of the Poisson process where each arrival is weighted according to a distribution. The compound Poisson process represents better the transactions dynamics. In this case, the assumption is that transactions sent to Cloudchain follow a Poisson process, but the amount of gas they require follows a compound Poisson process. The reason is that the difference between the amount of gas is based on the complexity of the transaction, for example, the creation of a contract requires a much higher amount of gas than updating the contract. Therefore, the probability of the required gas by X_j transactions occurring in $[0 - T]$ follows an exponential distribution based on the compound Poisson process as follows:

$$\mathbb{P}_j(X) = \frac{e^{-\omega T}(\omega T)^{X_j}}{X_j!} \tag{5.5}$$

5.4.1 Cloud Provider as a Requester

Here we explain the scenario from the perspective of CP_r that has to optimize its profit CPP_r while requesting VMs as follows:

$$\begin{aligned} CPP_r(G_r(t), D_r^{'}(t), t) \quad &= (p_r - \ \phi_r \ C_r) \ D_r + (p_r - p_s^{'}\phi_s^{'})D_r^{'}(t) \\ &- \frac{e^{-\omega T}(\omega T)^{X_r}}{X_r!} M_r^{'} G_r(t) + Rw_r\phi_r^{''} e^{-\frac{1}{\Gamma}M\eta} \end{aligned} \tag{5.6}$$

$M_r^{'}$ represents the amount of required gas that depends on the complexity of the transaction a provider wants to initiate. The transaction fees go to the miner that mines the block, so if a provider attends a mining process, it will be rewarded according to Eq.5.2 and Eq.5.4. $D_r^{'}(t)$ is the demand that CP_r intends to outsource to obtain the idle capacity of $\phi_s^{'}$ for a secondary price of $p^{'}$. Considering the time-dependent profit functions of CP_r in Eq.5.6, the objective function is the total discounted cloud provider's payoff over the planning horizon $[0 - T]$:

$$\begin{aligned} \text{maximize} \quad & \int_0^T e^{\rho t}\{CPP_r(G_r(t), D_r^{'}(t), t)\}dt \\ \text{subject to} \quad & \dot{D}_r^{'}(t) = G_r^{\beta_u}(t)\psi_r + \check{\theta} R_s^{\beta_u}(t) - \delta_r D_r^{'}(t) \\ & D_r^{'}(0) = D_{0r}^{'} \end{aligned} \tag{5.7}$$

The users' demands evolution over time is represented as $\dot{D}_r^{'}(t)$ for CP_r that increases when the service quality rises. The service quality is aggregated through two factors of gas price that CP_r pays and the reputation of CP_s. The demand decays at a certain rate of δ_r. It is important to note that Eq.5.7 formulates an optimal control problem with the gas price as a control variable and the cumulative demand of CP_r as a state variable. The analysis of differential games relies profoundly on the concepts and techniques of optimal control theory [35]. To study the dynamics of

the payoff function and the path of control variable, we leverage the Hamiltonian systems. Equilibrium strategies in the open-loop structures can be found by solving a two-point boundary value problem for ordinary differential equations derived from the Pontryagin maximum principle in Hamiltonian functions. The Pontryagin maximum principle gives the necessary condition for a control path to be optimal open-loop control. To acquire the optimal control, we first formulate the Hamiltonian system of the cloud provider's payoffs:

$$\begin{aligned} H_r(G_r(t), D_r^{'}(t), \lambda_r(t), t) = \; & (p_r - \; \phi_r \; C_r) \; D_r(t) + (p_r - p_s^{'}\phi_s^{'}) D_r^{'}(t) \\ & - \frac{e^{-\omega T}(\omega T)^{X_r}}{X_r!} M_r^{'} G_r(t) + R w_r \phi_r^{''} e^{-\frac{1}{\Gamma} M\eta} \\ & + \lambda_r(t)(G_r^{\beta_m}(t)\psi_r + \check{\theta} R_s^{\beta_m}(t) - \delta_r D_r^{'}(t)) \end{aligned} \tag{5.8}$$

According to the control theory, the optimal control strategy of the original problem must also maximize the corresponding Hamiltonian function. Thus, based on the Pontryagin maximum principle, the candidate optimal strategy has to satisfy the following necessary conditions:

$$\frac{\partial H_r(t)}{\partial G_r(t)} = -\frac{e^{-\omega T}(\omega T)^{X_r}}{X_r!} M_r^{'} + \lambda_r(t)\beta_m G_r^{\beta_m - 1}(t)\psi_r = 0 \tag{5.9}$$

$$\dot{\lambda}_r(t) = \rho\lambda_r(t) - \frac{\partial H_r(t)}{\partial D_r^{'}(t)} = (\rho + \delta_r)\lambda_r(t) - p_r + p_s^{'}\phi_s^{'}, \lambda_r(T) = 0 \tag{5.10}$$

When only one boundary condition is specified as $D_r^{'}(0) = D_{0r}^{'}$, the free-end condition is used as $\lambda_r = 0$ at $t = T$. The formulated differential equation Eq.5.10 can lead us

to the adjoint variable:

$$\lambda_r(t) = \frac{p_r - p_s^{'}\phi_s^{'}}{\rho + \delta_r}(1 - e^{(\rho+\delta_r)(t-T)}) \tag{5.11}$$

Replacing Eq.5.11 in Eq.5.9 gives us the optimal gas price control path as follows:

$$G_r^*(t) = (\frac{M_r^{'} e^{-\omega T}(\omega T)^{X_r}(\rho + \delta_r)}{X_r!(p_r - p_s^{'}\phi_s^{'})(1 - e^{(\rho+\delta_r)(t-T)})\beta_m \psi_r})^{\frac{1}{\beta_m - 1}} \tag{5.12}$$

5.4.2 Cloud Provider as a Supplier

Cloud provider as a supplier has a different scenario. CP_s observes the total demand of its own users, D_s, and the capacity preserved for the mining process to determine the remaining capacity $\phi_s^{'}$, to optimize its profit as follows:

$$\begin{aligned} CPP_s(R_s(t), D_r^{'}(t), t) \quad &= (p_s - \ \phi_s \ C_s) \ D_s + (p_s^{'} - \ \phi_s^{'} \ C_s^{'}) D_r^{'}(t) \\ &- \frac{e^{-\omega T}(\omega T)^{X_s}}{X_s!} M_s^{'} G_s(R_s(t)) + R w_s \phi_s^{''} e^{-\frac{1}{\Gamma} M \eta} \end{aligned} \tag{5.13}$$

$G(R_s(t))$ denotes the gas cost that the suppliers pay to earn higher reputation for having prompt communication. Considering the time-dependent profit functions of CP_s in Eq.5.13, the objective function is the total discounted cloud provider's payoff over the planning horizon $[0 - T]$:

$$\begin{aligned} \text{maximize} \quad & \int_0^T e^{\rho t} \{CPP_s(R_s(t), D_r^{'}(t), t)\} dt \\ \text{subject to} \quad & \dot{D}_r^{'}(t) = \hat{\theta}_r R_s(t)^{\beta_n} - \delta_s D_r^{'}(t) \\ & D_r^{'}(0) = D_{0r}^{'} \end{aligned} \tag{5.14}$$

The demand dynamics of CP_s is defined based on the demand that it receives from CP_r that evolves with its own reputation and decays at a rate δ_s. By solving

Table 5.2: Provider's estimated transactions and costs on Cloudchain based on the proposed scenarios

	Amazon EC2	Microsoft Azure	Rackspace	Century Link	Alibaba Cloud
Reputation Value	88	82	84	60	82
Price per hour (p)	0.0058	0.005	0.084	0.025	0.0125
Price per hour ($p^{'}$)	0.003	0.0025	n/a	n/a	n/a
Requests *	0	0	8	15	17
Supplies *	23	17	0	0	0
Cancellations *	0	0	0	3	2
Total Gas	1,290,668	953,972	15,292,736	34,310,286	36,254,668
Gas Price (gwei)†	15	15	15	12	11
Gas Cost (gwei) ‡	19,360,020	14,309,580	229,391,040	411,723,432	398,801,348
Gas Cost (USD) ‡	$12.06	$8.91	$142.91	$256.50	$248.45
Transaction Delay(s) §	27-66	27-66	27-66	27-4000	27-5459

*Quantity †Total Gas×Gas Price ‡Average §Time range of each transaction in seconds

a corresponding Hamiltonian system of Eq.5.14, similar to Eq. 5.8, the optimal reputation control path is obtained as follows:

$$R_s^*(t) = \left(\frac{M_s^{'} e^{-\omega T}(\omega T)^{X_s} G_s(\rho+\delta_s)}{X_s!(p_s^{'} - \phi_s^{'}\ C_s^{'})(1-e^{(\rho+\delta_s)(t-T)})\beta_n \hat{\theta}_r}\right)^{\frac{1}{\beta_n - 1}} \tag{5.15}$$

5.5 Implementation, Simulation and Discussion

We implemented the coopetitive Cloudchain prototype on Ethereum using Solidity (version 0.4.24), the script language on Ethereum, to test our proposed framework and the effect of gas price and reputation values on cloud providers revenues. This program is available open source in Github[3]. The program was written with the main concern of the minimum consumption of gas per each transaction and was tested using remix[4], an online IDE for Solidity.

The gas price unit is in gwei, which is 1×10^{-9} ether. Ethereum stores arbitrary

[3]https://github.com/kavehbc/Cloudchain
[4]http://remix.ethereum.org/

data in smart contracts in two ways. The first option is to store the data as a variable in a smart contract. The cost of storing data in the contract storage is based on the number of SSTORE operations on the contract variable. The second option is to store arbitrary data as a log event. There are also memory variables such as contract arguments and defined memory variables, which are not stored permanently inside the contracts. Memory variables are disposed after the function execution is complete. In our implemented prototype, we used solidity structures and variables to store provider's data and requests inside the contracts. Meanwhile, each transaction is logged with a summary using an event to make it easily accessible for the other providers (blockchain nodes) to track new transactions. Once a new transaction with a specific event (e.g. New Request) is created, other providers can call the contract to get more information and/or change contract stored data (e.g. to accept a new request). Calling a contract and retreiving data are expensive transactions, the stored data as events can provide enough information without any retreival cost. The events are retrieved and filtered using the Web3.js platform to notify the providers on important changes (e.g. New registration, updates, deactivations, new requests, etc.) in Cloudchain. CCR and CCP contracts are deployed once, but CCC would be deployed every time a new request is registered.

For the sake of representation, we assumed a small number of 5 cloud providers (Amazon, Microsoft, Rackspace, Alibaba cloud, and Century Link) using Cloudchain for a duration of 100 days to investigate their economic gain through the differential game. The scalability of our system for higher number of cloud providers is not questioned since the Ethereum platform is proven to be scalable. We simulated Rackspace, Alibaba and Century Link as cloud requesters who make 8, 17 and 15 requests of service, respectively. Meanwhile, Amazon accepts Rackspace and Century Links requests with a reputation threshold of 75, and Microsoft takes Alibaba's orders,

which were set for a minimum reputation of 85. Due to the limitation of Solidity in defining float numbers, we scaled the reputation values collected from Gartner to [0-100]. The on-demand cloud services' prices are borrowed from the providers' websites with an assumption of the secondary price of Amazon and Microsoft to be two times less for the Cloudchain members. The collected real-world data (e.g. reputation and price), simulated number of requests and supplies, as well as the simulated results of total gas consumption, gas price and transactions delays are shown in Table 5.2. Since there is no time-dependent profit maximization model similar to our proposal, not even in traditional centralized federations or related experiments to be compared to, only the results of our model are reported.

In our simulated scenario, three cloud providers of Amazon, Microsoft and Rackspace are supposed to be miners and collect their rewards. To make the simulation more realistic, we followed up all the contract transactions from registering in the Cloudchain up to confirmation of the contract completion, depositing the payment and assigning a reputation. Century Link and Alibaba are assumed to cancel their requests for few times after making the contract before acceptation. As Table 5.2 depicts, the obtained gas consumptions of cloud service requesters are much higher than those that answer these requests and supply these services. This is why Alibaba has the most and Microsoft the least gas consumption.

The gas price of Amazon and Microsoft are considered as constant inputs and they are set to 15. This price guarantees a fast execution of transactions to avoid tarnishing their reputation and will not impose them huge cost due to their minimal gas required as the role of suppliers. To estimate the time delay for each transaction, we tested different prices in different time slots to obtain an approximate range of delay depending on the traffic of the Ethereum network. The obtained optimal gas prices for the three cloud requesters are shown in Figure 5.2. Alibaba has to pay

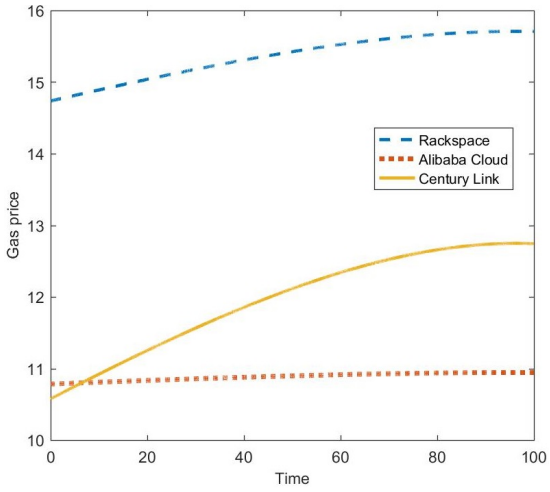


Figure 5.2: Gas prices of the three cloud service requesters

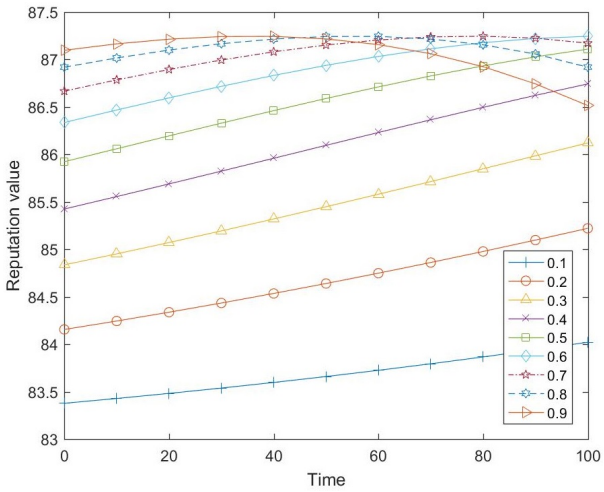


Figure 5.3: Microsoft's optimal reputation with different values of $(0.1 \leq \hat{\theta}_r \leq 0.9)$

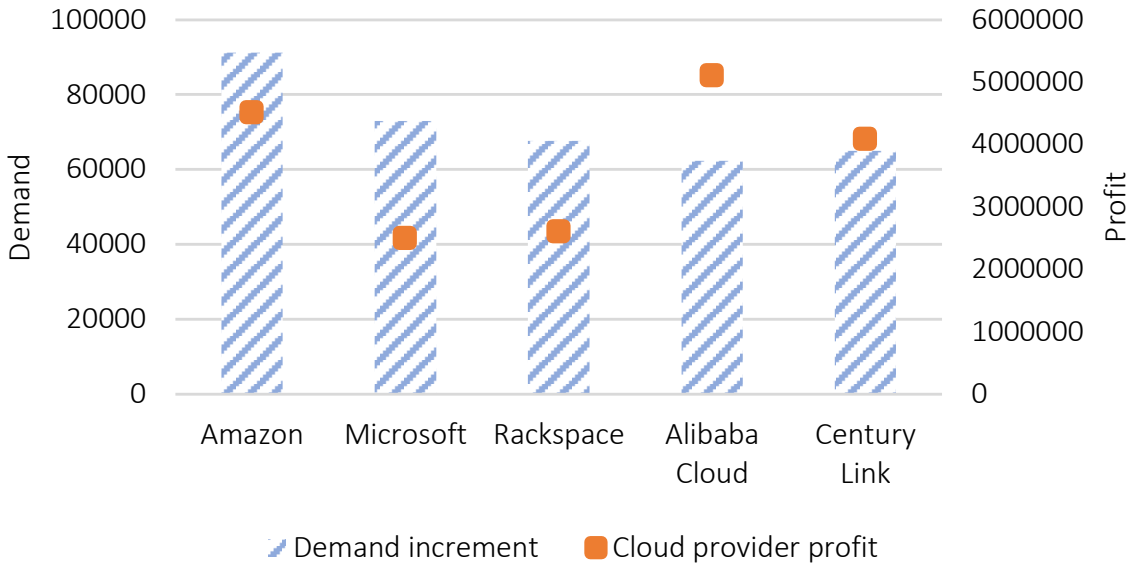


Figure 5.4: Average of cloud providers' profit and demands' evolution in Nash Equilibrium

the minimal price, which is almost 11 gwei for the whole period of time. This cloud provider has the highest number of requests, so it is not profitable if it invests more money over gas. With this price, Alibaba has to pay almost $248.45, at the time of writing this report. However, because of the cheap gas price, Alibaba has a delay of 27 to 5459 seconds for each transaction (refer to Table 5.2). Even though high traffic happens not very often, yet, it would be advisable to predict its demand in advance to avoid the delays that can cause user dissatisfaction. Century Link also has to pay cheap gas price, but not as cheap as Alibaba. It is reasonable since this cloud provider has less gas consumption, higher end-users' prices and lower reputation. To win the users' satisfaction proportional to its service's price, Century Link has to increase the gas price sharply, to speed up the communication and avoid major delays. Based on the results, Rackspace has to pay the highest price for the gas among the cloud requesters. The main reason can be the highest end users' price, the low amount of transactions and gas consumption. The participation in the mining process could also add up to its wealth to afford higher price and higher quality with minimum delays. It worth to note that even though the gas is costly for all cloud providers, it is a one-time cost for a permanent storage.

Figure 5.3 depicts Microsoft's optimal reputation value during these 100 days as obtained in our experiment. It is worth mentioning that Amazon showed a similar pattern. To investigate the behavior of cloud requesters' demand over these reputation values, we considered the demand rate $\hat{\theta}_r$ varying from 0.1 to 0.9. As the effectiveness of reputation over demands' rate raises, the provider has to aim for a higher reputation at the beginning to earn the eligibility for more demands. However, these optimums do not follow the same trend. In the case of lower effectiveness, the provider has to increase the service quality and gas price leading to a higher reputation over time, but as effectiveness gets intense, the reputation starts to decline. This is where the

provider has established its credibility at first and made the major profit halfway through the period, and the increase of reputation is not profitable anymore. This confirms that keeping a high reputation is costly and not always economically justified.

Figure 5.4 presents a comparative analysis of the average of profit and demands' evolution for the Cloudchain members. The demands' evolution $\dot{D}'_r(t)$ for cloud service suppliers have noticed a higher spike. Yet, interestingly, cloud service requesters have received a higher profit from Cloudchain due to fulfilling their initial demand and selling to their own end-users. The cloud service requesters could obtain cheaper prices from the suppliers and sell at their own prices. However, it should be noted that they can face the risk of not fulfilling their commitments to the end-users if none of the suppliers have the required preserved capacity to rent out. Although it seems that Cloudchain benefits more the cloud requesters, yet it is not true. The main profit of cloud suppliers is from their own market and users, and they only rent the partial idle computing resources, which are not being used. As the number of cloud service requesters elevates, their share of profit from the outsourced demand and the mining rewards increases.

5.6 Conclusion

In this chapter, we introduced a new distributed blockchain-based framework for cloud providers federation to overcome the limitations of conventional centralized federations. Due to the coopetitive environment of Cloudchain, and high expense of public smart contracts, we further designed and solved a differential game. This game modeled the best strategies of cloud providers to make a request with an optimal transaction cost and time, as well as, to optimize their reputation value to receive the requests from other providers. Cloudchain was implemented using Solidity over the

Ethereum network and the differential game was simulated for a sample of five cloud providers during 100 days. The findings can be summarized from two perspectives of the cloud service requesters and suppliers. For cloud requesters with a high number of requests, spending high gas price is not economically appealing. With cheaper gas prices, they might face some delays in peak times, which needs to be predicted in advance. Although requesters incurred higher costs from Cloudchain, yet they gained a significantly high income by outsourcing some parts of their customers' demands that could not be fulfilled by their own. The results showed that cloud suppliers have minimal gas consumption, which makes it more affordable for them to pay higher prices and enhance their communication and reputation. Though increasing the reputation was not always the best strategy for highly reputed cloud providers, a gradual increase is recommended when the requesters' demand is not significantly impacted. The end-user's service price is found to be a very decisive factor in deciding the level of quality and gas/reputation values for both of the cloud service requesters and suppliers.

Chapter 6

A Blockchain-based Model for Cloud Service Quality Monitoring

This chapter introduces a novel blockchain-based decentralized federation model that embodies quality verification for cloud providers who lease computing resources from each other. The blockchain structure removes the barriers of a traditional centralized federation and offers a fully distributed and transparent administration by enforcing the involved agents to maintain consensus on the data. For a blockchain-based federation, it is vital to avoid blind-trust on the claimed SLA guarantees and monitor the quality of service which is highly desirable considering the multi-tenancy characteristic of cloud services. Due to the fact that the blockchain network is unable to access the outside world, it cannot handle, by its own, providers misbehavior in terms of SLA violations. Thus, we introduce oracle as a verifier agent to monitor the quality of the service and report to the smart contract agents deployed on the

blockchain. Oracle is a trusted third-party agent who can communicate with the outside world of the blockchain network. The interaction between cloud service providers (either providing a service or requesting it from another provider) and the oracle through smart contracts comprises a system of autonomous and utility maximizer agents. Cloud requesters seek to receive high quality services with constant monitoring at cheap prices or even with no charge, while cloud providers aim to have a balanced work-load with less preserved capacity, and the oracle tends to charge higher for their monitoring services. Therefore, to model this conflicting situation, we formulate a dynamic Stackelberg differential game to optimize the cost of using the oracle and maximize the profit of the agents with the role of provider agent as leader, and the requester and verifier agents as followers. Our designed Stackelberg differential game can seize the dynamicity of users' demand and resource provisioning in a competitive cloud market. We implemented our proposed decentralized model using the Solidity language in the remix IDE on the Ethereum network. We further evaluated the optimal controls and agents' profit with real-world data simulated for three concrete cloud providers. The results revealed that the requester agent initiates most of the quality verification requests at the beginning to the middle time of the contract. Thus, the provider agent could reserve less computing resources considering the fact that it could share the workload among other customers' computing resources during the peak-time. Moreover, imposing higher penalty on the provider agent increased the capacity and decreased the number of requests for quality verification at the equilibrium. The evaluation also disclosed that the impact of timing in the dynamic pricing strategy of the verifier agent is very minimal, and the provisioning capacity of the provider is strongly correlated with the monitoring price. This chapter is published in [83].

6.1 Introduction

The demand variation has forced cloud providers to preserve a massive amount of computing resources to avoid SLA violation. To mitigate the issue of underutilized and over provisioned computing resources, cloud providers scaled their pool of resources by forming cloud federations to maximize their profit and provide guaranteed QoS [32]. In spite of their prominent advantages, cloud providers are reluctant to participate in federations due to some strict challenges, including the federations' stability, long-term commitments from the providers, fair revenue sharing, the presence of unknown and untrusted participants, security and privacy concerns regarding the managed data, and the creation and management overhead of these federations [4, 32, 47, 67]. In order to overcome the aforementioned limitations of the traditional federations, Cloudchain [82] proposed a new distributed blockchain-based framework to support interoperability and coopetition (i.e. cooperative competition) among the cloud providers. Cloudchain allows the cloud providers to outsource their unmet computing demands and agree on the values of shared variables (e.g. amount of the resource, SLA and price) and keep a history of how the values change over time.

Utilizing smart contracts in blockchain enabled Cloudchain to offer higher transparency, visibility, and reliance within its fully decentralized agreements deployed on top of Ethereum. However, Cloudchain falls short in supervising the SLA's agreed terms, which requires to access the outside world of the blockhcain network. Each of the cloud providers may disagree about the SLA compliance. However, investigating that is beyond the control of blockchain miners or digital codes embedded in the smart contracts due to its self-contained execution environment. Thus, we need a third party to perform the highly important verification task and confirm if the SLA is met.

To address the quality monitoring issue, we propose to employ oracles tailed

to smart contracts within an innovative multi-agent decentralized model. A smart contract is a piece of code deployed and executed on blockchain. Oracle in the blockchain context is a fully-trusted third-party agent that has access to the outside world, and feeds the data into the blockchain to be accessible by the applications. Oracles usually provide proofs to show that the retrieved data is tamper-proof. A number of oracles have been deployed using cryptographic evidences (e.g. hash code) such as Oraclize, or the Intel SGX feature, such as Town Crier, to make sure the data is tamper-proof. Our proposed multi-agent model includes five different agents, namely the cloud service requester (requester agent (RA)), cloud service provider (provider agent (PA)), oracle (verifier agent (VA)), and two smart contract agents called registry-profile and contract agents. These contracts that were developed in [82], are registered in the blockchain and are triggered by new transactions (i.e. initiating new requests or registering inputs from VA), which will make each blockchain node update its state based on the results obtained after running the smart contract. The smart contract is considered as an agent that has state variables and enforces the associated rules. In our scenario, RA makes a contract with PA and might initiate a quality monitoring request anytime from VA. VA can check the quality of the service with respect to different attributes (e.g. availability, bandwidth, response time, etc.) and detects any misbehavior of RA or PA, then returns the result to the contract agent to manage the payments and apply potential punishments. Accordingly, the contract agent decides who should pay the monitoring cost to VA.

Having cloud requesters, providers, and the oracle interacting with each other through smart contracts composes a system of autonomous and utility maximizer agents. Cloud requesters seek to receive high quality services with constant monitoring which could be very costly. On the other hand, providers aim to have a balanced work-load with less preserved capacity, yet avoid any monitoring cost or possible

punishments. If they do not manage the gap between the actual and ideal resource provisioning, it can negatively affect their reputation and aggregated utility. Meanwhile, the oracle tends to charge higher for the monitoring services without risking a decline in the number of the requests for monitoring that it receives. Yet some important questions remain: how many times and when to ask for quality monitoring, who has to pay for such a service, how much should be paid and how to avoid SLA violations and its possible consequences. To answer the above questions and to optimize the providers' computational resource capacity, quality verification requests and cost of the monitoring, we formulate a dynamic Stackelberg differential game among three agents seeking to maximize their revenue. The Stackelberg differential game is used to study the sequential decision making of cloud provider (leader) for the optimal resource provisioning, cloud requester (follower) for the quality monitoring requests, and oracle (follower) for the monitoring cost. In the designed game, the differential equations capture the dynamic competition and resource provisioning, quality monitoring requests and costs in continues time.

This study contributes as follows:

1. Developing a novel blockchain-based decentralized model for cloud providers that outsource some parts of their demand which they cannot fulfill on their own. Our proposed model enjoys a multi-agent structure, which allows us to introduce a quality verifier agent to ensure the cloud provider's compliance with the SLA. The interaction of an oracle within blockchain for monitoring purposes is innovative.

2. Formulating a three-player dynamic Stackelberg differential game in which players have to make choices about their control variables at various points in time, where PA acts as the leader and RA and VA are the followers. Differential equations are introduced into the game model to characterize the dynamic

variations of the end users' demand. Finally, the optimal solutions are obtained based on the open-loop equilibria of the proposed game.

We evaluate our proposed model using the Solidity language on Ethereum and Web3.js by simulating three real-world cloud providers using our system for 100 days. To the best of our knowledge, there is no research that implements oracles and their practical integration with smart contracts. Due to the very recently emerging research topic and nonexistence of any similar model, we are not able to compare our model with any other model. In addition to the optimal profit of agents, we also evaluated estimated transactions and costs.

6.2 Motivational Scenario

For a blockchain-based federation, it is vital to monitor the QoS and ensure that SLA conditions are met, since cloud providers may have an incentive to deviate. This verification is highly desirable considering the multi-tenancy characteristic of cloud services. In this context, to scale the economic benefits and optimize resource utilization, multiple VMs are initiated on the same physical server simultaneously. The performance variation depends on the network load and usage peak from other tenants. Cloud providers try to balance the workloads and achieve the required performance with less preserved capacity, yet they might not be able to supply a consistent performance.

Figure 6.1 illustrates a scenario when cooperation among two cloud providers could be problematic. Let us imagine cloud provider A and cloud provider B are using Cloudchain through the following steps:

1. Cloud providers have to supply scalable cloud services with consistent performance for their users with guaranteed SLA. To ensure such a scalability

and on-promise performance, cloud providers A and B can register themselves in Cloudchain to enjoy the federated services from the available resources.

2. When cloud provider B faces a computing resources deficiency to meet its end users' demand, it can create a request through Cloudchain.

3. Provider A who has idle servers, accepts the request and leases its computing resources to provider B within the smart contract deployed over Cloudchain with a specified SLA, price, terms and conditions.

4. Two issues might happen that Cloudchain cannot resolve on its own. First, provider A has actually complied with SLA stated in the Cloudchain contract, but provider B claims falsely that provider A violated the SLA conditions and has to be fined.

5. Cloudchain is impotent to oversight and confirm who is telling the truth due to its inability to communicate with the outside world of the blochchain network. Blockchain can only access information present in a transaction or in the transaction history of the blockchain. Thus, we introduce oracle as a third party to perform the verification and confirm if the SLA is met.

6. The oracle can check the QoS at some cost and report the SLA compliance to Cloudchain.

7. Second issue can arise when provider A has actually compromised the quality but denies the accusation and requests to receive the full deposit from provider B.

8. Cloudchain calls the oracle and initiates a verification request.

9. The oracle confirms that a violation has happened.

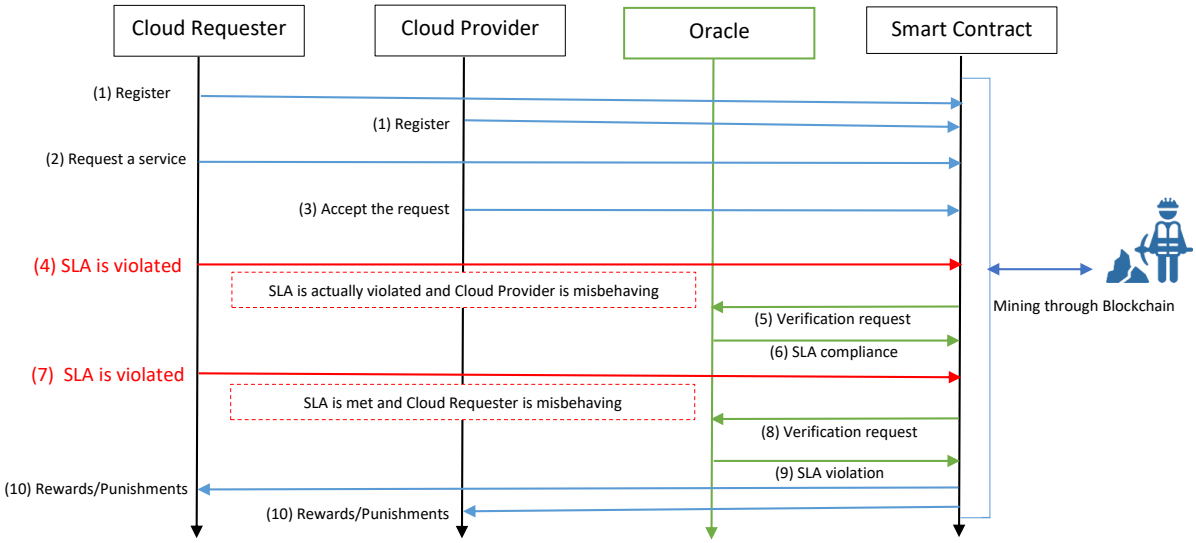


Figure 6.1: Cloud providers misbehavior through the federation

10. At the end, considering the terms and conditions of the contract, as well as the verification reports from the oracle, Cloudchain distributes the money and charges the verification cost.

Utilizing the oracle through the steps 4 to 9 is part of our contribution in this paper. In order to fully materialize the oracle as a verifier agent, we first develop a new multi-agent structure and then optimize the cost of using the oracle and trading within Cloudchain.

6.3 Related Work

This work extends our previous work on Cloudchain [82], a novel model that exploits blockchain to prompt and support interoperability and coopetition among cloud providers over the public Ethereum network. Blockchain is employed to ensure transparency and decentralize the agreements in Cloudchain, but the provided cloud services are supplied out of the blockchain network. So, the blockchain dynamics can neither guarantee nor validate the quality of the supplied service. Therefore,

to ensure that the providers comply with the agreements, we need to validate the QoS, even though the agreement is deployed over blockchain. Similar to any other blockchain-based platform, Cloudchain suffers from the most challenging issue, yet to be solved, which is its inability to interact with the outside world. Thus, in this work, we introduce a verifier agent as an autonomous oracle to monitor the quality upon the request of the service requester (RA). We further investigate the revenue maximization strategies among the cloud providers and verifier agent which were not discussed in our previous contribution.

Summaries of the related literature are drawn from three different areas elaborated as follows.

6.3.1 Game Theory in Cloud Computing

Game theory has been successfully applied in the cloud computing area, for instance for resource allocation and pricing mechanisms, where the interactions of players have to be taken into account [65]. A user-provider interactive approach is taken by Hadji et al. [31], where a Stackelberg game is designed to consider constrained pricing with limited resources offered by a cloud service provider and the optimal user demands. Xu et al. [91] optimized a pricing policy for cloud service providers to better compete with each other under the evolution of the cloud market. Forming a Stackelberg game, the authors applied a reinforcement learning (Q-learning) to find out an optimal policy for the leader. Following the leader, the optimal policy for followers will be uncovered. However, the price is the only utility factor considered in these studies and the importance of QoS is somehow neglected. The study by Fan et al. [24] could address the QoS competition issue by considering the market competition among a SaaS (software-as-a-service) provider and a traditional software provider as a differential game. This research analyzes short and long-term competitions for price

and dynamic quality between the two firms. The authors found that the cost of software implementation can significantly affect the equilibrium price while quality improvement has a more robust effect.

6.3.2 Cloud Federation

The literature about cloud providers cooperation focuses on federation formation as coalitional games where capacity and revenue are shared [63]. Coronado et al. had an intensive investigation on federation formation variables among providers, including revenue sharing mechanisms, capacity and cost disparity, and the presence of a big competitor [17]. They defined revenue sharing mechanisms as the most important factor. Among these mechanisms, shapely value and outsourcing models had the least and best performance, respectively. They indicated that collaborating providers can implement a mechanism in which a provider outsources some of its business and gets a percentage of the revenue. The outsourcing model allows the provider to keep some of the revenue of its secured business, even though it is not able to fulfill that business alone. The authors had an insight through the demand peaks and concluded that cloud providers tend to stay in outsourcing collaboration when the demand is high. However, interoperability, trust among providers, and service quality or SLA are not considered in their study. The findings from this work confirm the superiority of outsourcing in terms of maximizing the profit of providers, which is what we are proposing in this paper in addition of having the advantage of coopetition among these providers. The fact that providers tend to collaborate when they face a hike in their demand, reinforces the consideration of a dynamic and long/short-term federation-like blockchain. The challenges of interoperability and trust issues among cloud providers are also addressed by the blockchain platform we propose in this paper.

Wahaabb et al. [87] focused on the business potential of Web services and

addressed the problem of community-based cooperation as a virtual trading market using a Stackelberg game model. They further developed a trust-based hedonic coalitional game model that forms trusted communities of cloud services and proposed an algorithm to converge to a stable coalition [88]. Their simulations proved the importance of federation and showed an improvement to the Nperformance of the established communities in terms of availability, throughput and response time. Khosrowshahi [2] considered stability and fairness for all web services within a community and offered an applicable mechanism for membership requests and selection of web services. The proposed mechanism used cooperative game-theoretic techniques, particularly Shapley value, core, ϵ-core and convex games. Nonetheless, none of these studies utilized blockchain to form a federation and neither proposed a solution for service quality monitoring for federated services.

Zhao et al. [103] investigated the impact of two factors: energy consumption and SLA violations on degrading the cost-efficiency of data centers and the cloud providers' revenue. The authors developed online VMs placement algorithms as an optimization problem of maximizing revenue from VMs migration and achieved promising results. However, no initiatives are proposed to monitor the SLA violations, specifically when it come to cooperation and competition among providers. The dynamic and timed decision making strategies are also not considered.

6.3.3 Blockchain and its Applications

Blockchain is emerged as a distributed database technology building upon a secured list of timestamped transaction records. Its main innovation stems from enabling parties to transact with untrusted parties over a computer network [53]. The blockchain data structure is an ordered list of blocks containing aggregated transactions. Every block is identifiable and linked to the previous block in the chain

where the integrity is ensured by cryptographic techniques. Recently, blockchain had a revolutionary impact in corporate governance by offering greater transparency among stakeholders, easier administration, and creation of an infrastructure for innovative applications where business transactions could be shared in real time [96]. By leveraging blockchain-enabled smart contracts, we eliminate the need for trust in federation and reduce barriers of entry, lock-in, and transaction costs, by removing obsolete trust-establishing mechanisms [43]. A smart contract is a piece of code residing on a blockchain and is identifiable by a unique address. Moreover, smart contracts permit creating decentralised applications (DApp) that operate autonomously without any intervention by a system entity.

A few efforts have been made to study the potential of blockchain in real-world applications despite its great potential for businesses to share data and collaborate in a secure and customized manner [53]. According to Tractica, a market research firm, the annual revenue for enterprise applications of blockchain is estimated to reach $19.9 billion by 2025 [39]. The majority of studies about blockchain's application have focused on finance [85], energy [60] and IoT applications [102].

In cloud computing and service industry, to the best of our knowledge, there have been very few related academic initiatives in addition to Cloudchain. Among which, one paid a major attention in the energy-aware resource management problem in cloud datacenters and developed a robust blockchain-based decentralized resource management framework in order to save the energy consumed by the request scheduler [92]. Moreover, this research further utilizes a reinforcement learning embedded in a smart contract to minimize the energy cost. Their simulations based on Google cluster traces and electricity prices showed their method was able to reduce the datacenters' cost significantly. Desmaa [43] is a cloud marketplace framework based on the blockchain technology. This conceptual framework modeled the interactions

between a service provider and a service consumer and tried to overcome problems of conventional marketplace systems, such as barriers of entry and transaction costs. Yet, the outsourcing model with collaboration and competition among cloud providers themselves are not considered in this initiative. Moreover, the providers' profit and the best strategies for utilizing this marketplace are not elaborated nor modeled. Even though the authors developed a prototype, no evaluation and validation against real-world's scenarios were provided.

6.4 Quality Verification Model within Cloudchain

6.4.1 Background: Cloudchain's Smart Contracts

Cloudchain incorporates three types of smart contracts including a set of executable functions and state variables [82]. Similar contracts are proposed in [3] in the context of medical data management. *Contract 1* (*C1*) or Cloudchain Registry (CCR) is a global contract that maps cloud providers identification values (including *Name*, *Reputation Value*, *Computing Capacity* and *Storage Capacity*) to their Ethereum address identities (equivalent to public keys). Policies coded into the contract can regulate registering new providers or changing the mapping of the existing ones. The cloud provider registration can be restricted only to certified providers. CCR also maps identities to the Cloudchain Contract (CCC) address on the blockchain, where a special contract regarding each provider profile and list of services is recorded.

Contract 2 (*C2*) denotes Cloudchain Profile (CCP). It holds a list of references to CCC, representing all the participants' previous and current engagements with other nodes in the system. CCP also implements a functionality to enable provider notifications. Ethereum supports an event-based mechanism which permits smart contracts to create an event and signals that a certain action (e.g. an update

to profile's data) has been performed. Providers should register their requests in the CCP contract to be propagated and raised to other nodes of providers. Each transaction list stores a status variable. This indicates whether the transaction is newly established, awaiting pending updates and has or has not been completed. This contract is important as it stores the address of all new CCC contracts, without which Cloudchain can simply lose the track of all the contracts.

Contract 3 (*C3*) represents the Cloudchain Contract (CCC). It is issued between two nodes in the system when one node accepts and provides the requested service for the other. The beneficiaries can also complete or cancel the contract. Once the contract is completed or canceled, the contract balance would be transferred to the supplier or requester address respectively, and the contract status would also be updated. There are two approaches to reduce the size of the data as well as the cost of transactions over Cloudchain. The first approach is a common practice for data storage in smart contracts and consists of storing raw data off-chain, and meta-data, small critical data, and hashes of the raw data on-chain [94]. However, selection of off-chain data storage has some concerns regarding the interaction between the blockchain and the off-chain data storage. The other approach is to provide a common glossary among cloud providers to define the generic terms and policies to be referred to in the contract. The members of Cloudchain can join and leave the system anytime by executing specific functions in the smart contracts. Such a flexible membership allows them to supply or demand a service once or multiple times as required.

6.4.2 Multi-Agent Architecture of the Cloud Service Distributed Model

The proposed model incorporates five agents as follows:

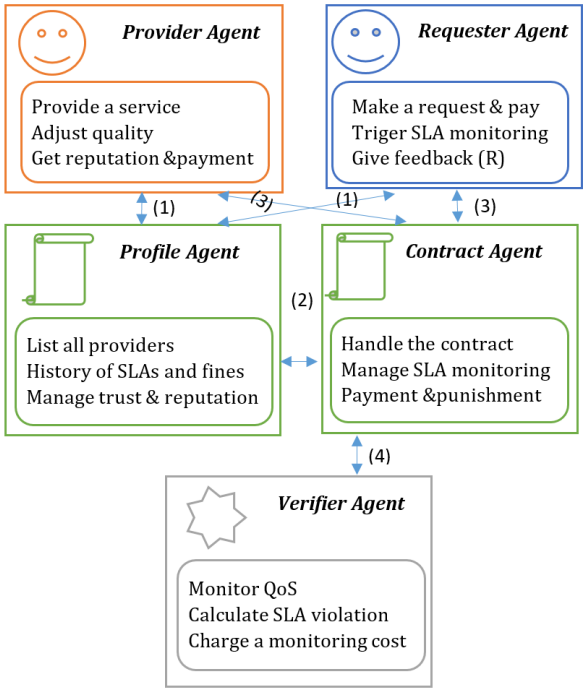


Figure 6.2: Multi-agent cloud service quality monitoring model

- *The Requester Agent (RA)* and *Provider Agent (PA)* are both cloud providers willing to trade their computing resources;
- *The Registry-Profile Agent (RPA)* and *Contract Agent (CA)* represent CCR-CCP and CCC smart contracts, respectively, with a set of executable functions and state variables; and
- *The Verifier Agent (VA)*, known as oracle, is an agent that verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts.

Figure 6.2 provides the interactions among these agents. In step 1, RA and PA should register themselves in RPA where each registered user is assigned with a public key pair. RPA maps identities of RA and PA to the contract agent's (CA) address on the blockchain. It holds a list of references to CA to provide all the participants' previous

and current engagements with other nodes in the system with a record of any SLA violation or compliance.

When RA faces a computing resources deficiency, it can submit a request for a service using RPA to create and deploy a CA in the blockchain in step 2. Meanwhile, a rule for providers is set by the requesters to ensure that qualified providers could ultimately receive the task, e.g. reputation value threshold. The first time reputation values can be computed from the customers' ratings given to each provider through online rating platforms and will be updated based on the future ratings given by RA.

CA regulates the interactions between two nodes in the system where one node accepts and provides the requested service to the other in step 3. RA is required to pay a deposit in advance and it is stored in the contract using CA. The beneficiaries can complete or cancel the contract, however, the contract termination and delivery of the requested service have to be confirmed by RA. Once completed or canceled, CA calculates fines to be charged if any exist and the remained contract's balance would be transferred to the RA or PA address respectively, and the contract status would be updated.

Function calls on contracts are transactions, and those which update the contract storage need to be validated by blockchain miners. Once a new block is mined with the newly linked CA, it would be broadcasted to other nodes and the first node that accepts the request should update the respective CA contract.

RA can initiate a quality monitoring request anytime to check if the provider is complying with the SLA conditions during the runtime. The request should be submitted to CA and CA calls VA to perform the verification through step 4. VA checks the prioritized quality attributes of the service using RA credentials and extracts the runtime needed information and pushes it into CA. PA would be penalized if there is any violation of the SLA. A record of the SLA monitoring and penalties

will be kept by RPA for future references. RA is required to rate the supplier based on the perceived performance. The process of requesting a service and monitoring its quality among the five agents are elaborated in Algorithms 6.1 and 6.2. The agents' decision variables and the details of their trading policies provided in these algorithms will be discussed in the following section.

Algorithm 6.1 Cloud providers service agreements within the multi-agent model

Require: Ether Deposit; Reputation threshold; PA reputation; Cloud requester's Etheruem address (RA); Cloud supplier's Etheruem address (PA).

procedure ServiceAgreement
    RA makes a service request in RPA
    RPA creates a CA
    RA.SendTo(CA, Ether Deposit)
    CA.Availability = True
    EventLog.Create("New request is available")
    if RA.Reputation $\geq$ Reputation threshold AND RA.Accept(CA) **then**
        CA.Availability = False
        while RA requests a quality verification **do** ▷ refer to Eq.6.8
            CA calls ***Algorithm 6.2***
            $N(t)$ += 1
            if Verification.Result = True **then**
                RA.SendTo(Verifier, $M(t)$)
                PA.PositiveVerification += 1
            else if Verification.Result = False **then**
                PA.SendTo(Verifier, $M(t)$)
                PA.NegativeVerification += 1
                Assign F to charge PA
                CA.SendTo(RA, F)
            end if
        end while
        if CA.Completed **then**
            EventLog.Create("CA is completed")
            ContractDeposit = CA.TotalAmount
            CA.SendTo(PA, ContractDeposit)
            EventLog.Create("Fund is transferred to the Cloud supplier")
            RA.UpdateReputation(PA)
        end if
    end if
end procedure

Algorithm 6.2 Verification process

Require: CA Ethereum address; $M(t)$; ϵ; VA Etheruem address (oracle).
Ensure: Verification.Result ▷ Boolean

```
1: procedure VerificationProcess
2:     VA retrieves CA terms and monitor the cloud service
3:     VA verifies the quality of the provided service based on CA.SLA
4:     if (φ̄ − φ(t)) <= ϵ then
5:         Verification.Result = True
6:     else
7:         Verification.Result = False
8:     end if
9:     Calculate M(t) to charge CA                    ▷ refer to Eq.6.15
10: end procedure
```

6.5 Requester, Provider and Verifier Agents Decision Making

Unlike conventional (static) game theoretic models, dynamic models we use in this paper consider the important dimension of time and recognize the competitive decisions that do not necessarily remain fixed. Models involving competition in continuous time are typically treated as differential games, in which critical state variables, e.g. demand, are assumed to change over time according to specified differential equations.

Our RA, PA, and VA agents aim to maximize their profit within our proposed multi-agent and blockchain-based federation. RA is facing a peak time and is going to request some VMs from other federation's members, and PA has some idle servers and is willing to rent them out with the price offered by RA. For simplicity and without losing generality, we will focus on a single VM type, with $\bar{\phi}$ denoting the desired capacity and the process rate of VM instances that can be hosted by PA which guarantees to meet the SLA even during the peak time. In fact $\bar{\phi}$ is what RA is paying for, while ϕ is the actual preserved capacity that PA assigns for RA considering the fact that its other customers might use less and it can assign the extra capacity

to RA when needed. Therefore, PA controls its optimal capacity assignment ϕ. Since RA might experience a QoS degradation or sense a violation of SLA, it has the right to initiate a monitoring verification request. However, this request can be costly as it has to pay if there is no SLA violation. Thus, RA is required to decide on the number of verification requests to make, denoted by the control path $N(t)$ (a control path is a vector of control or decision variables). On the other hand, VA has to decide on its control path representing the optimal pricing $M(t)$ which alters its verification demands and final revenue.

We formulate the profit maximization, service trading, and quality assurance problems as a Stackelberg differential game as follows.

- Players: There are three players: PA acts as leader; and VA and RA act as followers. We assume that the decision making of the followers are simultaneous and they use each others' control variable as input of their models.

- Strategy space: PA can choose the optimal capacity control path $\phi(t)$ to maximize its payoff by observing the cost and numbers of the requests for monitoring in response to the capacity. VA sets an optimal price control path $M(t)$ to charge for the quality verification by considering the given capacity which also affects the number of the requests from RA. RA controls its verification requests $N(t)$ to ensure the service quality by considering the given $\phi(t)$.

- State: The end users' demands and the quality monitoring demands are the system states of PA, RA, and VA, respectively.

In the Stackelberg differential game, both the leader and followers try to maximize their own payoffs, which are the integration of instantaneous payoffs over the time horizon $[0, T]$, by controlling their control paths which are their decision variables to

be committed to for the whole time horizon. Similar to a statistic Stackelberg, to obtain the equilibrium of a Stackelberg differential game, we use backward induction and solve the problem for the follower first. A list of the notations is provided in Table 6.1.

Table 6.1: Notations used in Stackelberg differential game

j/i	Index of a provider/quality attribute from the set k/I
M	Quality monitoring cost
N	Number of monitoring verification requests
$r/p/v$	Requester/provider/verifier agents
G/C	Cost of gas/capacity
F	Fine payment of PA to RA due to SLA violation
W	The amount of the cumulated gas for each block
$W^{'}$	The amount of required gas for each transaction
X	Number of the transactions over the blockchain
Y	Maximum number of the initiated verification requests
ω	Rate of transactions arrival for an agent in $[0-T]$
$\omega^{'}$	Weight of the monitoring request arrival from RA
Γ	Rate of the block generation for miners
$\phi/\phi^{'}/\bar{\phi}$	Provider's active/mining/maximum capacity
P	Price per VM
R	Reputation value of RA and PA in the range of $[0-1]$
Rw	Reward value of mining
τ	Block propagation time
η	Rate of the impact of W over τ
ψ	Rate of end users' demands increase due to higher quality for RA
θ	Rate of end users' demands increase due to higher R_p
β	Quality sensitivity of the end users
γ	Price sensitivity of RA and PA for monitoring service
δ	Demand decay rate
μ	The amount of VMs
Q	Quality attributes of a cloud service in the set I
$\lambda/\Lambda/\Theta$	Adjoint variables

6.5.1 Preliminaries

This section explains some of the required elements to formulate our game. We first make an assumption that is a rule for differential games.

Assumption: Each player has perfect knowledge of:

- The dynamic state function determining the evolution of the demand, and the control paths of the three players.
- The payoff functions.
- The initial demand states at time zero.

The analysis of differential games relies profoundly on the concepts and techniques of optimal control theory [35]. Definition 4.1 provides some relevant points on this regards.

Axiom 6.1. The open-loop strategy spaces of RA, VA and PA are respectively defined as:

$\tilde{N} = \{N(t)|N(t)$ measurable on $[0,T], N(t) \geq 0$ for all $t \in [0,T]\}$

$\tilde{M} = \{M(t)|M(t)$ measurable on $[0,T], M(t) > 0$ for all $t \in [0,T]\}$

$\tilde{\phi} = \{\phi(t)|\phi(t)$measurable on $[0,T], 0 > \phi(t) \geq \bar{\phi}$ for all $t \in [0,T]\}$. The strategy profile $(N^*(t), M^*(t), \phi^*(t))$ is an open-loop Stackelberg equilibrium if, for RA, VA and PA, each of them is optimal control strategies given others' strategies.

It is assumed that cloud providers can participate in the mining to earn some rewards if they have spare computing resources. In the mining race, miners have to compete to solve proof of work and propagate the block to reach consensus. The new blocks' generation follows a Poisson process with a constant rate $\frac{1}{\Gamma}$ throughout the whole blockchain network [45]. Before the race, miners collect their selected pending transactions into their blocks with a total gas amount of

$\sum_{j=1}^{k} W_j$. Gas is a proportional amount that Ethereum pays to motivate the miners to participate in the mining process. When miner j propagates its block to the blockchain for consensus, the time for verifying each transaction is affected by the total size of the transactions W_j. The first miner j who successfully has its block achieves consensus will be rewarded based on the amount of the assigned capacity $\phi_j^{'}$. The amount of the reward can be computed by the following proposition.

Theorem 6.1. *The miner j's expected reward* $Rw_j(\phi_j^{'})$ *is:*

$$Rw_j(\phi_j^{'}) = Rw_j\phi_j^{'}e^{-\frac{1}{\Gamma}W_j\eta_j} \tag{6.1}$$

Proof. The expected reward of mining is: $Rw_j\mathbb{P}_j(\phi_j^{'}, W_j)$ where $\mathbb{P}_j(\phi_j^{'}, W_j)$ is the probability that miner j receives the reward by contributing a block. To win the reward, the miner must perform a successful mining and instant propagation. However, the miner may fail to obtain the reward if its new block does not achieve consensus as the first. This kind of mined block that cannot be added to the blockchain is called orphaned block. The block containing a larger size of transactions has a higher chance of becoming orphaned since a larger block requires more propagation time, thus, causing a higher delay for consensus. As the arrival of new blocks follows a Poisson distribution, miner j's orphaning probability, $\mathbb{P}_j^0$, can be approximated as:

$$\mathbb{P}_j^0 = 1 - exp(-\frac{1}{\Gamma})\tau_j \tag{6.2}$$

It is safe to assume that miner j's block propagation time τ_j is linear with the size of transactions in its block, $\tau_j = W_j\eta_j$, where η_j is a constant that reflects the impact of

W_j over τ_j. Therefore, we obtain the reward probability from Eq.6.2 as follows:

$$\mathbb{P}_j(\phi_j^{'}, W_j) = 1 - \mathbb{P}_j^0 = \phi_j^{'} e^{-\frac{1}{\Gamma} W_j \eta_j} \tag{6.3}$$

By multiplying Eq.6.3 by the reward value, we obtain Eq.6.1. □

Axiom 6.2. To model the transactions' distribution, we use the compound Poisson process, which is a generalization of the Poisson process where each arrival is weighted according to a distribution. The compound Poisson process represents better the transactions dynamics. The assumption is that transactions sent to the blockchain follow a Poisson process, but the amount of gas they require follows a compound Poisson process. The reason is that the difference between the amount of gas is based on the complexity of the transaction, for example, the creation of a contract requires a much higher amount of gas than updating the contract. Therefore, the probability of the required gas by X_j transactions occurring in $[0 - T]$ follows an exponential distribution as follows:

$$\mathbb{P}_j(X) = \frac{e^{-\omega T}(\omega T)^{X_j}}{X_j!} \tag{6.4}$$

We further require another distribution function to formulate the cost and penalty/reward of the quality monitoring services provided by VA. To do so, we require to consider how the history records of transactions and the reputation values of the providers can influence the number of the requests for quality verification. This formulation is provided in the below definition.

Axiom 6.3. Let us define Y as the maximum number of the verification requests that an agent j can initiate. If the reputation value of an agent who is providing a service is high, it is likely that there will be less number of the verification requests. However, if the reputation value of RA is high it is more likey to have more numbers

of verifications to ensure that quality will remain reliable and the reputation stays untouched. Thus, the most requests will be initiated if the reputation value of RA is high and PA is low. We propose a similar distribution function given by Eq.6.4, since the initiation of monitoring requests and the amount of cost $M(t)$ or the received reward (penalty F paid by PA) follows the same compound Poisson process which is tight with the agents' reputation values:

$$\mathbb{P}_j(Y) = \frac{e^{-\omega' T}(\omega' T)^{Y_j(1-R_p)R_r}}{(Y_j(1-R_p)R_r)!} \tag{6.5}$$

6.5.2 Problem Formulation and Open-Loop Equilibrium of RA (Follower)

We firstly discuss the problem of the optimal number of verification requests $N(t)$ for RA, to get the open loop equilibrium solution. Adopted from [79], we define the end users' demand using the Cobb-Douglas function that captures the demand elasticities and variations specific for each user of a cloud service, $D = \mu\ P^{-\alpha}\ Q^{\beta}$. The two variables α and β are the users' sensitivities towards the price and quality, respectively.

As presented in Algorithms 6.1 and 6.2, we assume that if $(\bar{\phi} - \phi(t)) <= \epsilon$, then PA is complying with the SLA and there will be no penalty of F and no verification cost $M(t)$ for PA. Here, ϵ is a very small number that PA is allowed to disobey due to the very dynamic context of cloud service attributes. But, if $(\bar{\phi} - \phi(t)) > \epsilon$, then RA will not pay for $M(t)$ and will be rewarded the penalty value of F. The provider's cost is usually considered to be quadratic in the literature [33], a convex cost term can prevent aggressive behavior of a certain provider which can result in a monopoly market.

The requester agent acts as a follower of PA by relying on the provided capacity

and simultaneously receiving the monitoring cost from VA to adjust its optimal number of the verification monitoring requests $N(t)$, while considering the evolve of its end users' demand state. Thus, it tries to maximize its profit according to the following function:

$$\begin{aligned} \text{Maximize} \quad & RA(N(t), D_r(t), M(t), \phi(t), t) = \\ & \int_0^T e^{\rho t}\{P_r D_r(t) - \bar{\phi} P_p^2 + Rw_r(\phi_r^{'}) - \mathbb{P}_r(X) W_r^{'} G_r \\ & - \mathbb{P}_r(Y)(M(t) - F)(\bar{\phi} - \phi(t))N(t)\}dt \\ \text{subject to} \quad & \dot{D}_r(t) = (N(t)\phi(t))^{\beta}\psi - \delta_r D_r(t) \end{aligned} \tag{6.6}$$

$Rw_r(\phi_r^{'})$, $\mathbb{P}_r(X)$ and $\mathbb{P}_r(Y)$ are borrowed from Proposition 4, Definition 4.2 and Definition 4.3. ρ denotes discounted factor in our discounted-utility model, in which it is assumed that the instantaneous utility each period depends solely on profit in that period, and that the utilities from streams of profit are discounted exponentially.

The demand state dynamics is defined with $\dot{D}_r(t)$ which increases with the number of capacity monitoring and ensuring the service quality with the power of β as the end user sensitivities towards the quality at the rate ψ. It also decays at the rate δ_r in which users switch to another provider.

Axiom 6.4. For RA, the number of monitoring requests' strategy $N(t)^*$ is optimal if the following inequality holds for all feasible control $N^*(t) \neq N(t)$, $RA(N(t)^*, D_r(t)) \geq RA(N(t), D_r(t))$.

In order to get the equilibrium solution of the optimization problem in Eq. 6.6, we need to construct the Hamiltonian system of the RA's problem. Equilibrium strategies in the open-loop structures can be found by solving a two-point boundary value problem for ordinary differential equations derived from the Pontryagin maximum principle in Hamiltonian functions. Here, the equilibrium solution for RA

is the solution of the differential game, and also is the Stackelberg equilibrium solution for RA as a follower. The Hamiltonian system of RA is as follows:

$$\begin{aligned} H_r(t) =& P_r D_r(t) - \bar{\phi} P_p^2 + R w_r(\phi_r^{'}) - \mathbb{P}_r(X) W_r^{'} G_r \\ & - \mathbb{P}_r(Y)(M(t)r - F)(\bar{\phi} - \phi(t))N(t) + \\ & \lambda(t)((N(t)\phi(t))^{\beta}\psi - \delta_r D_r(t)) \end{aligned} \tag{6.7}$$

The adjoint variable or shadow price (λ) associated with a particular constraint is the change in the optimal value of the objective function per unit increase in the right-hand-side value of that constraint, all other problem data remaining unchanged. The economic interpretation of $\lambda(t)$ is the value of an additional unit of the end users' demand for RA. For a given $N(t)$, $\lambda(t) > 0$ implies that RA benefits from the current demands. With a zero shadow price $\lambda(t) = 0$, RA does not take into account the impact of the price on future users' demands. On the other hand, when $\lambda(t) < 0$, RA has no motive to sacrifice current profits for future profits by paying the cost of quality monitoring, so that it will no longer increase $N(t)$. The final solution is obtained in the following Theorem.

Theorem 6.2. *Knowing the fact that verification cost is paid by RA unless there is a violation of the SLA by PA, in which PA incurs $M(t)$ and F, the optimal number of monitoring requests is given by:*

$$N(t)^* = \begin{cases} \left(\frac{-\mathbb{P}_r(Y)M(t)(\bar{\phi}-\phi(t))}{\lambda(t)\beta\psi\phi(t)^{\beta-1}}\right)^{\frac{1}{\beta-1}} & \text{if } (\bar{\phi} - \phi(t)) <= \epsilon \\ \left(\frac{\mathbb{P}_r(Y)F(\bar{\phi}-\phi(t))}{\lambda(t)\beta\psi\phi(t)^{\beta-1}}\right)^{\frac{1}{\beta-1}} & \text{if } (\bar{\phi} - \phi(t)) > \epsilon \end{cases} \tag{6.8}$$

and $\lambda(t)$ is given by:

$$\lambda(t) = \frac{P_r}{\rho - \delta_r}(1 - EXP((\rho - \delta_r)(t - T)) \tag{6.9}$$

Proof. As proven in the optimal control theory, the optimal control strategy of the original problem must also maximize the corresponding Hamiltonian function. According to the Pontryagin's Maximum Principle (PMP), a control constitutes an open loop equilibrium to the problem in Eq. 6.7, and $D_r(t)$ is the corresponding state trajectory, if there exists a costate function $\lambda(t)$ such that the following relations are satisfied,

$$\begin{aligned}\frac{\partial H_{RA}(t)}{\partial N(t)} =&\mathbb{P}_r(Y)(M(t)r - F)(\bar{\phi} - \phi(t))+ \\ &\lambda(t)\beta\psi(N(t)\phi(t))^{\beta-1} = 0\end{aligned} \tag{6.10}$$

$$\dot{\lambda}(t) = \rho\lambda(t) - \frac{\partial H^*_{RA}(t)}{\partial D_r(t)} = \lambda(t)(\rho - \delta_r) - P_r, \lambda(T) = 0 \tag{6.11}$$

where Eq.6.11 is the adjoint equation to describe the dynamics of a costate variable. In the case that the strategy space $\tilde{N}$ does not depend on the system state D_r, the maximized Hamiltonian function H^*_{RA} on the right hand side of Eq.6.11 can be replaced by the original H_{RA}. When only one boundary condition is specified as $D_{r0}(t) = D_r(0)$, the free-end condition is used as $\lambda = 0$ at $t = T$. Solving the differential equation of Eq.6.11 can lead us to the corresponding costate function. By solving Eq.6.10, we can obtain the optimal $N(t)^*$ given in Eq.6.8. □

6.5.3 Problem Formulation and Open-Loop Equilibrium of VA (Follower)

To formulate the optimal pricing control $M(t)$ problem for VA and to get the open loop equilibrium solution, we require to define the dynamic variation of the state of verification demand. A major part of dynamic pricing research originates from the Bass new service diffusion model, which was later enriched by incorporating price sensitivity to allow a dynamic pricing examination [30, 69]. We modify this model to

elaborate on the new concept of verification demand in our model as described in the following definition.

Axiom 6.5. Let $V(t)$ denotes the total number of the verification requests initiated by RA at time t for each quality attribute, given an I-tuple of QoS attributes $Q_1, Q_2, ..., Q_I$ with an index of i. The verification state evolves based on the external factor of capacity discrepancy and the internal factor of price as follow:

$$\dot{V}(t) = dV(t)/dt = (M(t)V(t) + (\bar{\phi} - \phi(t)))(1 - \gamma M(t)) \tag{6.12}$$

where the positive parameter γ measures the providers' sensitivity to the verification price.

To obtain a suitable dynamic pricing strategy, VA observes the number of the verification requests from RA and provides a response to the announced capacity control of PA. Therefore, VA tries to maximize its profit by the following Eq. 6.13 which is subject to Eq. 6.12:

$$\begin{aligned} \text{Maximize} \quad & VA(M(t), N(t), V(t), t) = \\ & \int_0^T e^{\rho t}\{(M(t) - C_v^2)V(t) - \mathbb{P}_v(X)W_v^{'}G_v\}dt \\ \text{subject to:} \quad & \dot{V}(t) = (M(t)V(t) + (\bar{\phi} - \phi(t)))(1 - \gamma M(t)) \end{aligned} \tag{6.13}$$

The Hamiltonian system is given as below.

$$\begin{aligned} H_{VA}(t) =& (M(t) - C_v^2)V(t) - \mathbb{P}_v(X)W_v^{'}G_v + \\ & \Lambda(t)(M(t)V(t) + (\bar{\phi} - \phi(t)))(1 - \gamma M(t)) \end{aligned} \tag{6.14}$$

Theorem 6.3. *The optimal monitoring cost is given by,*

$$M(t)^* = \frac{1}{\gamma} - \frac{(\bar{\phi} - \phi(t))}{V(t) + \Lambda(t) + 1} \tag{6.15}$$

where $\Lambda(t)$ *is given by:*

$$\Lambda(t) = (\frac{\gamma(\bar{\phi} - \phi(t)) + V(t) - C_v^2 V(t)\gamma}{V(t)(\rho\gamma + 1)})(1 - EXP(\rho + \frac{1}{\gamma})(t - T)) \tag{6.16}$$

Proof. Similarly, the necessary optimality conditions for VA can be derived according to PMP as follows:

$$\begin{aligned} &\frac{\partial H_{VA}(t)}{\partial M(t)} = 0 \\ &V(t) + \Lambda(t)(V(t)(1 - \gamma M(t)) - \gamma(M(t)V(t) + (\bar{\phi} - \phi(t)))) = 0 \end{aligned} \tag{6.17}$$

By solving Eq.6.17, we can obtain the optimal price given by Eq.6.15. When the optimal control depends on the system state, it has to be replaced in the original Hamiltonian system in Eq.6.14 to achieve $H_{VA}^*(t)$ and to be used for calculation of the adjoint variable $\Lambda(t)$.

$$\begin{aligned} &\dot{\Lambda}(t) = \rho\Lambda(t) - \frac{\partial H_{VA}^*(t)}{\partial V(t)}, \Lambda(T) = 0 \\ &\dot{\Lambda}(t) = \Lambda(t)(\rho + \frac{1}{\gamma}) - \frac{\bar{\phi} - \phi(t)}{V(t)} - \frac{1}{\gamma} + C_v^2 \end{aligned} \tag{6.18}$$

□

6.5.4 Problem Formulation and Open-Loop Equilibrium of PA (Leader)

For each capacity path $\phi(t) \in \tilde{\phi}$ PA announces, there is a corresponding $N^*(t) \in \tilde{N}$ and a $M^*(t) \in \tilde{M}$. PA takes the VA and RA's best responses into consideration when

solving the optimization problem. The PA's optimization problem is given by:

$$\begin{aligned}&\text{Maximize } PA(\phi_s(t), R(t), D_s(t), \lambda(t), \Lambda(t), \Gamma(t), M^*(t), N^*(t), t)\\&= \int_0^T e^{\rho t}\{P_p D_p(t) - \phi(t)C_p^2 + Rw_{jp}(\phi_p^{'}) - \\&\mathbb{P}_p(X)W_p^{'}G_p - \mathbb{P}_p(Y)(M(t)^* + F)(\bar{\phi} - \phi(t))N(t)^*\}dt\end{aligned} \tag{6.19}$$

$$\text{subject to} \quad \begin{cases} \dot{D}_p(t) = (R_p - F(\bar{\phi} - \phi(t)))^{\beta}\theta - \delta_p D_p(t) \\ \dot{\lambda}(t) = \lambda(t)(\rho - \delta_r) - P_r \\ \dot{\Lambda}(t) = \Lambda(t)(\rho + \frac{1}{\gamma}) - \frac{\bar{\phi} - \phi(t)}{V(t)} - \frac{1}{\gamma} + C_v^2 \end{cases} \tag{6.20}$$

Compared to RA and VA, the Hamiltonian function of PA in the Stackelberg differential game is more complex since the maximization of the payoff of PA also needs to consider the dynamics of costate variables of RA and VA as the additional state constraints besides the system state constraints. In this case, similar to the introduction of a costate variable for the system states in the follower's Hamiltonian function, the costate variables for both the system states and costates of the followers are needed in the Hamiltonian function of PA as leader.

$$\begin{aligned}H_{PA}(t) =& P_p D_p(t) - \phi(t)C_p^2 + Rw_{jp}(\phi_p^{'}) - \mathbb{P}_p(X)W_p^{'}G_p - \\&\mathbb{P}_p(Y)(M(t)^* + F)(\bar{\phi} - \phi(t))N(t)^* + \\&\Theta_1(t)(\theta(R_p - F(\bar{\phi} - \phi(t)))^{\beta}) - \delta_p D_p(t)) + \\&\Theta_2(t)(\lambda(t)(\rho - \delta_r) - P_r) + \\&\Theta_3(t)(\Lambda(t)(\rho + \frac{1}{\gamma}) - \frac{\bar{\phi} - \phi(t)}{V(t)} - \frac{1}{\gamma} + C_v^2)\end{aligned} \tag{6.21}$$

Similarly, the necessary optimality conditions for PA can be derived through the PMP. Due to the concavity of Hamiltonian function with respect to $\phi(t)$, we can obtain $\phi^*(t)$ for the leader which could be obtained from $\frac{\partial H_{PA}(t)}{\partial \phi(t)} = 0$, and denoted as:

$\phi^*(t) = g_p(M^*(t), N^*(t), \lambda(t), \Lambda(t), \Theta_1(t), \Theta_2(t), \Theta_3(t), D_p(t), V(t), t).$

We further have the following conditions:

$$\begin{aligned}
&\dot{\Theta}_1(t) = \rho\Theta_1(t) - \frac{\partial H_{PA}(t)}{\partial D_p(t)} = \Theta_1(t)(\rho - \delta_p) - P_p \\
&\dot{\Theta}_2(t) = \rho\Theta_2(t) - \frac{\partial H_{PA}(t)}{\partial \lambda(t)} = -\Theta_2(t)\delta_r - \\
&(\tfrac{1}{1-\beta})\lambda^{\frac{\beta}{1-\beta}}(\tfrac{\mathbb{P}_j(Y)F(\bar{\phi}-\phi^*(t))}{\beta\psi\phi^*(t)^{\beta-1}})^{\frac{1}{\beta-1}}\mathbb{P}_p(Y)(M^*(t)+F)(\bar{\phi}-\phi(t)) \\
&\dot{\Theta}_3(t) = \rho\Theta_3(t) - \frac{\partial H_{PA}(t)}{\partial \Lambda(t)} = \\
&\rho(\Theta_3(t)-1) + \frac{1}{\gamma} + \frac{\mathbb{P}_p(Y)(\bar{\phi}-\phi^*(t))^2 N(t)^*}{V(t)+\Lambda(t)+1}
\end{aligned} \tag{6.22}$$

with boundary conditions $\Theta_1(T) = 0$ and $\Theta_2(0) = \Theta_3(0) = 0$. We impose $\Theta_1(T) = 0$, because $D_r(T)$ is free to move and impose $\Theta_2(0)$ and $\Theta_3(0)$ to be 0, because our problem is controllable and initial state depends on $\phi(0)$. Replacing $\phi^*(t)$ into Eq.6.16 along with differential equations in Eq.6.22 constitutes a system of five differential equations which, along with the boundary conditions, imply a solution; however is difficult to obtain analytical solutions for that (you can refer to [22] for a discussion of the complexity of the solutions to a similar system). Yet, we analyze all the variables and the system behavior in Section 5.

Theorem 6.4. *For the formulated Stackelberg differential game, the candidate strategy profile* $(N^*(t), M^*(t), \phi^*(t))$ *is indeed an open-loop Stackelberg equilibrium.*

Proof. It is straightforward that the construction of strategy profile $(N^*(t), M^*(t), \phi^*(t))$ satisfies all the necessary conditions as it followed the PMP conditions. The following arguments constitute the sufficient conditions for optimality. Since the Hamiltonian function H_{RA} is strictly concave and continuously differentiable with respect to $N(t)$ for all $t \in [0, T]$, the necessary optimality condition in Eq.6.10 uniquely determines a candidate optimal control path $N(t)^*$ as

a function of the observed verification pricing strategy $M(t)$, capacity strategy $\phi(t)$ and the system state $\dot{D}_r(t)$, and the costate $\lambda(t)$. In a similar way, due to the strict concavity of Hamiltonian function H_{VA} with respect to $M(t)$, and H_{PA} with respect to $\phi(t)$, PMP provides not only necessary conditions but also sufficient conditions for optimality of $M^*(t)$ for VA and $\phi^*(t)$ for PA. According to the stated conditions, we can conclude that the obtained strategy profile is indeed an open-loop Nash equilibrium. □

6.6 Implementation and Evaluation

We implemented our proposed blockchain-based quality monitoring prototype on Ethereum using Solidity (version 0.4.25), the script language on Ethereum and Web3.js. This program is available open source in Github[1]. The program was written with the main concern of the minimum consumption of gas per each transaction and was tested using remix[2], an online IDE for Solidity. The gas price unit is in gwei, which is 1×10^{-9} ether. In our implemented prototype, we used solidity structures and variables to store provider's data and requests inside the contracts. Meanwhile, each transaction is logged with a summary using an event to make it easily accessible for the other providers (blockchain nodes) to track new transactions. Once a new transaction with a specific event (e.g. new request) is created, other providers can call the contract to get more information and/or change contract stored data. To make the simulation more realistic, we followed up all the contract transactions from registering up to confirmation of the contract completion and assigning a reputation.

For the sake of representation, we assumed three real-world cloud providers (Amazon (PA), Alibaba cloud (RA), and Century Link (RA)) using the system for

[1] `https://github.com/kavehbc/Cloudchain`
[2] `http://remix.ethereum.org`

a duration of 100 days to investigate their economic gain through the Stackelberg differential game. The scalability of our system for a higher number of providers is not questioned since the Ethereum platform is proven to be scalable. We simulated Alibaba and Century Link as cloud requesters who make 17 and 14 requests of service, respectively. The on-demand services' prices are borrowed from the providers' websites and their ratings are collected through the Gratner's dataset[3]. The collected real-world data, simulated number of requests and simulated results of total gas consumption, gas price, gas cost (at the time of writing this paper), and transactions delays are shown in Table 6.2.

The gas price that providers choose to pay for each transaction can affect the speed of processing their transactions to be approved since miners choose the most profitable transactions to include in their block. We adopt the optimal gas price formulated in our previous study [82]. As Table 6.2 depicts, the obtained gas consumptions of cloud service requesters are much higher than those that answer these requests and supply these services. This is why Alibaba has the most and Microsoft the least gas consumption. To estimate the time delay for each transaction, we tested different prices in different time slots to obtain an approximate range of delay depending on the traffic of the Ethereum network. Since there is no time-dependent profit maximization model similar to our proposal, not even in traditional centralized federations or related experiments to be compared to, only the results of our model are reported.

Figure 6.3.a illustrates the optimal $\phi(t)$ for Amazon, where $\bar{\phi} = 304$ according to the speed attribute mentioned in the SLA terms of Amazon. It can be easily noticed that during the first half period of T, it is crucial to preserve a capacity close to the desired capacity all the time, otherwise, Amazon incurs a huge loss. The surprising

[3]`https://www.gartner.com/reviews/market/public-cloud-iaas`

Table 6.2: Provider's estimated transactions and costs based on the proposed scenarios

Cloud providers	Amazon	C-Link	Alibaba
Reputation	0.9	0.6	0.8
Price	0.0058	0.025	0.0125
Requests	n/a	14	17
Consumed gas *	1,739,596	32,022,933	36,254,668
Gas Price	15	12	11
Gas cost (G) †	26,093,940	384,275,203	398,801,348
Gas Cost (USD) ‡	$12.06	$256.50	$248.45
Transaction Delay(s) §	27-66	27-4000	27-5459

* $\sum_{Y=1}^{y} W'$ †Total Gas×Gas Price ‡Average §Time range of each transaction in seconds

point is that Amazon can cheat over the preserved capacity after $t = 60$, since it does not influence its profit. In this situation, Amazon will not reserve the whole resource for Century Link, and if the request consumes extra computing resource than the reserved one, the workload will be shared with other tenants. In this way, there is a minimal risk of penalty and monitoring surcharge as the number of the Amazon's customers and tenants grow over time. It should be noted that the pattern was similar for both followers.

The optimal pricing of VA in response to $\phi(t)$ is provided in Figure 6.3.b. According to our findings, the impact of timing in dynamic pricing is very minimal, meanwhile capacity is strongly correlated with monitoring price. As the capacity increases, the computational cost and time for VA also climb. Consequently, VA has to enhance its price to be profitable. Another reason could be the low number of the monitoring requests, initiated when the capacity is almost desired, as shown in Figure 6.3.c. Century Link had the most number of the requests during the first half of the period with a very high intensity at the beginning where $\phi(t)$ was low. For the second half period, Century Link is well informed about the quality and the results of the verification from VA, so the number of the requests is almost flattened. This

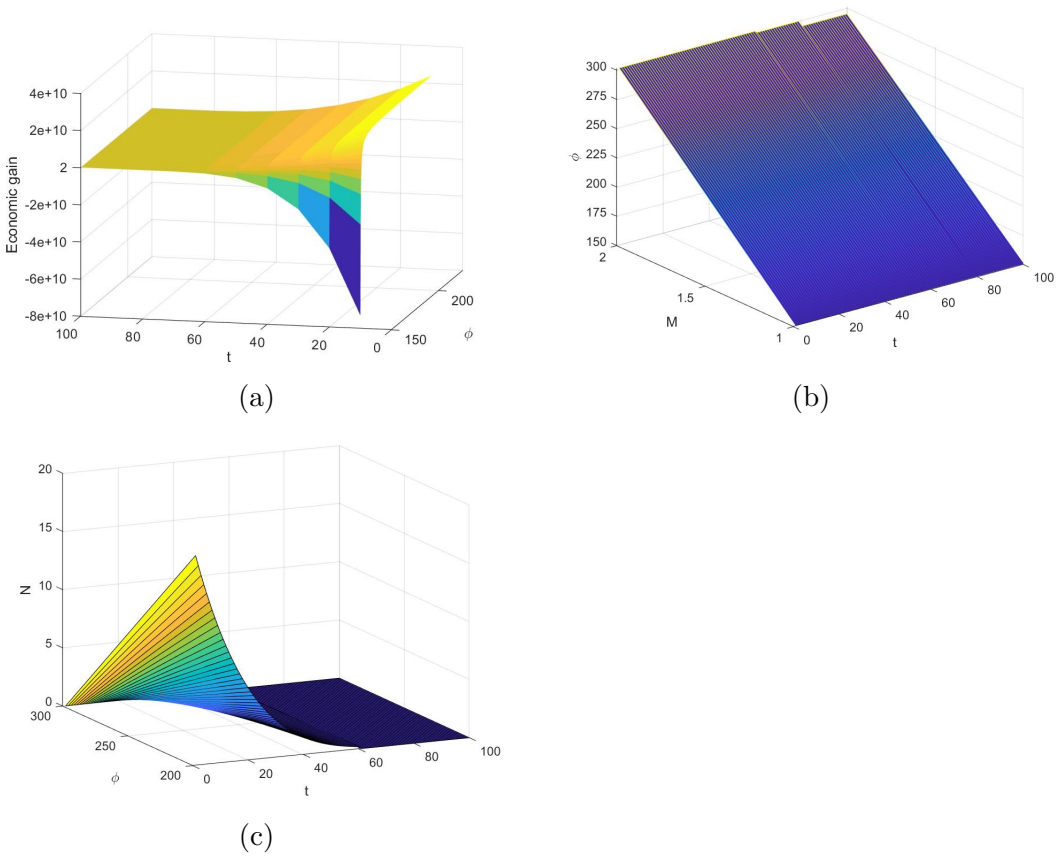


Figure 6.3: (a) Optimal capacity of Amazon (PA) emerging towards equilibrium, (b) Dynamic pricing strategy of VA in the equilibrium, (c) Optimal quality verification requests for Century Link (RA) in the equilibrium

behavior of Century Link now justifies why Amazon can cheat over the quality after $t = 60$. Alibaba Cloud also showed a similar pater, though the scale was different. Alibaba had a higher number of requests for verification due to its higher reputation value and number of transactions. It worths to mention that this response is given to a finite time, it could be different if we assume they will be collaborating for infinite time.

We further investigated the effect of the penalty value and the reputation value of PA over RA's profit and optimal control. As shown in Figure 6.4, the number of the requests for verification starts declining unexpectedly, as penalty F for Amazon

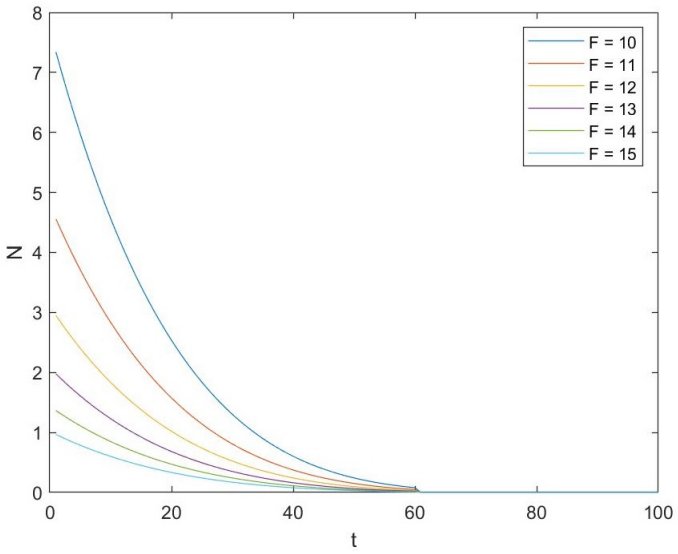


Figure 6.4: Optimal quality verification requests for Alibaba (RA) in the equilibrium

(meaning reward for Alibaba) is getting higher. This means that with a higher penalty, Amazon will not risk over $\phi(t)$, so the probability of earning a reward is low. Whereas, it is pretty much probable that Alibaba ends up with the monitoring cost to be paid. So, imposing higher penalty to the provider agent, will increase the capacity and decrease the quality verification requests' equilibria. The reputation value of PA has a significant effect on the profit of RA. As shown in Figure 6.5, Alibaba with a reputation of 0.8 gain most if the reputation of PA is higher than Alibaba itself. If the reputation value of Amazon drops to less than 0.8, it is not economically justified to outsource Alibaba's demand to it. This highlights the effect of the users' satisfaction over the demand evolution and economic gain over time. The reputation threshold is certainly less for Century Link with lower reputation value.

6.7 Conclusion

To overcome the issue of traditional federations of cloud providers and compromised QoS, this research proposed a multi-agent blockchain-based quality monitoring model.

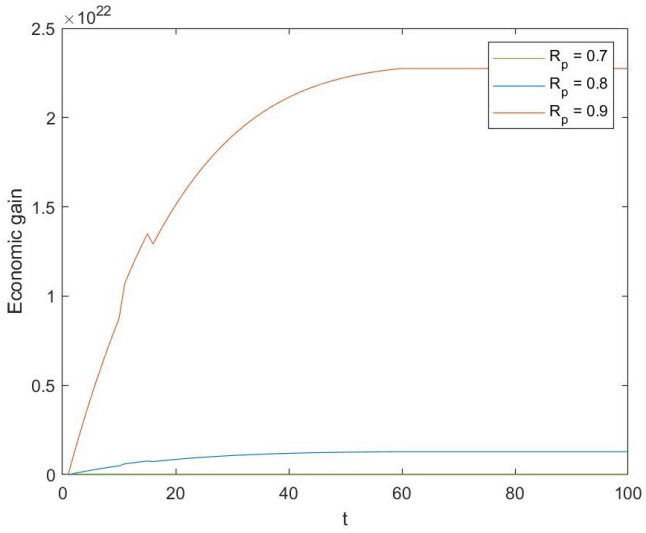


Figure 6.5: The impact of the Amazon's reputation over Alibaba's profit

In our proposed model, a multi-agent approach was taken where an oracle plays the role of a verifier agent to evaluate the service quality whenever is called through the smart contract agents deployed on the blockchain. A Stackelberg differential game was designed to formulate the best strategies of resource provisioning, the number of quality verification requests and the monitoring price for the provider, requester and verifier agents, respectively. The system was implemented using Solidity on Ethereum and was simulated for resource trading among three real-world cloud providers. It was found that at the beginning, the provider has to preserve the desired amount of capacity to satisfy the required quality even throughout the peak-times. However, it is not economically justified to make this reservation for the last periods of its contracts' time. Such a resource provisioning impacted the verifier pricing strategy and the number of requests. RA asks for more verification during the first half period of the contract when the preserved capacity is low. The reputation of PA elevated the profit of RA, whereas, it negatively affected the reputation of PA when it is lower than RA. Furthermore, higher penalty raised the capacity and reduced the number of verification requests at the equilibrium. The developed system was proven to be

economical for cloud beneficiaries and valuable in transparency and preventing the SLA violation.

Chapter 7

BLOR: Bayesian Bandit Learning Model for Blockchain Oracles Reliability

Smart contracts struggle with the major limitation of operating on data that is solely residing on the blockchain network. The need of recruiting third parties, known as oracles, to assist smart contracts has been recognized with the emergence of blockchain technology. Oracles could be deviant and commit ill-intentioned behaviors, or be selfish and hide their actual available resources to gain optimal profit. Current research proposals employ oracles as trusted entities with no robust assessment mechanism, which entails a risk of turning them into centralized points of failure. The need for an effective method to select the most economical and rewarding oracles that are self-interested and act independently is somehow neglected. Thus, this chapter

proposes a Bayesian Bandit Learning Oracles Reliability (BLOR) mechanism to identify trustless (a term used in the context of blockchain systems meaning not requiring trust) and cost-efficient oracles. Within BLOR, we learn the behavior of oracles by formulating a Bayesian cost-dependent reputation model and utilize reinforcement learning (knowledge gradient algorithm) to guide the learning process. BLOR enables all the blockchain validators to verify the obtained results while running the algorithm at the same time by dealing with the randomness issue within the limited blockchain structure. We implement and experiment with BLOR using Python and the Solidity language on Ethereum. BLOR is benchmarked against several models where it proved to be the most efficient algorithm in selecting the most reliable and economical oracles with a fair balance.

7.1 Introduction

Blockchain technology has the ability to cut the role of middlemen by enabling self-enforcing digital contracts (called smart contracts), whose execution does not require any human involvement in a safe, secure, and immutable way. The emergence of the blockchain as a revolutionary technology has been compared to the Internet, and it has predicted that it will erode power from centralized authorities. With its deployment as a service [50] and its integration with IoT [68], Blockchain has a promising approach in supporting business collaborations by ensuring transparency to all the stakeholders if conflicts arise [8,37]. However, the integration of blockchain with external data is one of the major obstacles preventing widespread adoption. Imagine that two persons place a bet on who wins a football match and deposit their funds in a smart contract. Based on the results of the game, the smart contract should release the funds to the winner. However, a smart contract does not have access to the data

out of its network and should ask a trusted party to learn who won the match.

In blockchain, the term oracle refers to an entity that can access external data without compromising the integrity of the blockchain. Oracles are assumed to be third-party agents that are trustworthy and can communicate with the outside world, and fetch the data into the blockchain [93]. Oracles are also able to connect the blockchain to external databases. This way, costly computations can be carried out outside of the blockchain. Oracles ensure the integrity of the retrieved data by providing some evidences [44]. Thus, cryptographic-based evidences such as the ones used by Oraclize[1], or trusted hardware-based evidences such as the ones used by the Town Crier system that leverages Intel SGX [99] are used as part of a number of oracle-based systems. These evidences are not only insufficient to ensure that the data is tamper-proof, they are impractical in many real-world applications where the digital data is not available or human involvement is required.

Oracles could display ill-intentioned behaviors, or unable to perform their tasks due to lack of capacity and being selfish by failing to report their real available resources [49]. Thus, placing a reliable mechanism to select the right oracles plays a significant role in a blockchain network's success. There are several proposals for organizing one or more oracles as a group with trustworthy mechanisms, specifically designed for computer hardware and software [28], [6]. However, these methods are not applicable when human intervention is involved or when the original data source is malicious. Moreover, these proposals sought to organize one or more oracles with enhanced security features or incentive mechanisms. To the best of our knowledge, there is no smart mechanism to promote how to select the most rewarding oracles among the existing ones in a market of oracles that might act selfishly to gain optimal profit.

[1] `https://provable.xyz/`

In this research, we utilize a Bayesian multi-armed bandit to learn the most rewarding oracles from the two perspectives of reliability and cost efficiency, to perform specific tasks within a blockchain. Multi-armed bandit is a reinforcement learning method that assumes the player does not know how much it will earn each time playing a particular slot machine, but the player has a distribution of belief, which could be wrong. The only way the player learns who has the highest expected reward is to try all machines, even those that do not appear to be the best. While trying these machines, the player may be earning lower rewards. The ultimate goal is to balance what we earn against what we learn (to improve future decisions) to maximize the expected sum of rewards. In our case, oracles are considered to be slot machines and blockchain beneficiaries are players who try to recruit the best oracles. Reinforcement learning methods have been applied in many real-world applications and their employment within blockchain has great advantages including high accuracy, ability to learn with few or no historical record, and low computational resources consumption [77]. To the best of our knowledge, these methods have not been applied in the field of blockchain yet, and even though it would be very interesting and novel, serious challenges in design and implementation within current platforms arise.

Theoretical and practical challenges: The issue of selecting the most rewarding oracle is a decision-making problem that should capture the tensity between exploration of new oracles and exploitation of the good and well-known ones. For simple and low number of choices, dynamic programming can compute the optimal solution. However, it is very computationally inefficient in the blockchain environment with the growing number of oracles working for blockchains. There is a need for an algorithm that runs quickly with a very minimal computation surcharge. The reason is that this algorithm has to be running by all blockchain validators (i.e., miners) acting within the network. Furthermore, current solutions of multi-armed bandit

assume that the player retains little information about the past, or switch between exploration and exploitation either randomly or after a fixed number of trials. These solutions are not practical for our problem, since oracles could be run and managed by intelligent agents that can change their behavior anytime. Another challenge of utilizing current solutions is that our decision-making procedure should be based not only on the oracles' performance, but also on their cost of performing the task considering applications' limited budgets. There could be some reliable and high performance oracles that are expensive, but current solutions would always select them based on their past performance records. We assume a fixed cost for each oracle, and consider the oracles reputation and cost of other oracles in the market could change the behavior of each individual oracle.

To overcome the aforementioned challenges, we formulate a Bayesian cost-dependent reputation model to learn the behavior of oracles and utilize knowledge gradient algorithm which guides the learning process based on the marginal value of information. Using a Bayesian model for blockchain is complex, since the algorithm has to produce the same results in every course of experiment. This is because all the validators should verify the results and it only happens if all of them come up with the same results while running the algorithm. This adds further complexity since all the Bayesian reinforcement learning methods include randomness and use random variables. At last, the current platforms of blockchains and smart contracts are very limited, for example no floating number can be defined within blockchain, or limited number of variables can be defined for Ethereum. This paper discusses how the proposed model and mechanism tackles and solves these issues by formalizing the oracles' performance optimization as a Bayesian bandit problem. Our algorithmic model defines a distribution over oracles with different reputations (representing their reliability and costs) to be used by blockchain participants to choose best performing

oracles on future requests.

Contributions: This chapter contributes as follows:

1. Formulating a new model using a Bayesian cost-dependent reputation model (BCRM) and knowledge gradient (KG) to find the most rewarding oracles. BCRM captures the behavior of the oracles elegantly, and KG unfolds the exploration/exploitation dilemma in multi-armed bandit with very low computational cost and high accuracy.

2. Proposing a framework to show how to employ the model within a blockchain where all the validators need to achieve a consensus. This framework incentivizes oracles to continuously act honestly and provide a fair balance of quality and price with minimal possibility of acting maliciously.

3. Adapting a reinforcement learning algorithm for blockchain environment with limited computational resources and capabilities (e.g., there is no floating number in Ethereum). Designing and implementing a reinforcement learning solution for the oracle selection problem is an objective yet to be achieved.

We simulated and implemented our proposed model using Python on Google Colab and Solidity on Ethereum. The implementation of BLOR deals with many challenges raised by the complexity of machine learning and limitations of blockchain and Ethereum, such as floating numbers, randomness and advanced mathematical numbers that are not supported in blockchain. Since there is no real-world data on oracles working for blockchains, we had to simulate the behavior of 100 oracles during 1000 observations to assess the performance of our model and compare it with other comparative algorithms.

7.2 Motivational Scenario: Trust Paradox of Oracles and Blockchains

Many blockchain platforms have been experiencing the oracle idea since the beginning of Ethereum, but the oracle dilemma continues unsolved at a large scale. The most challenging part is that majority of oracles require a level of trust, which directly opposes the trustless blockchains' nature. The main complication of using oracles is trusting them as outside sources of information. The trust issue connected with oracles is referred to as the oracle problem.

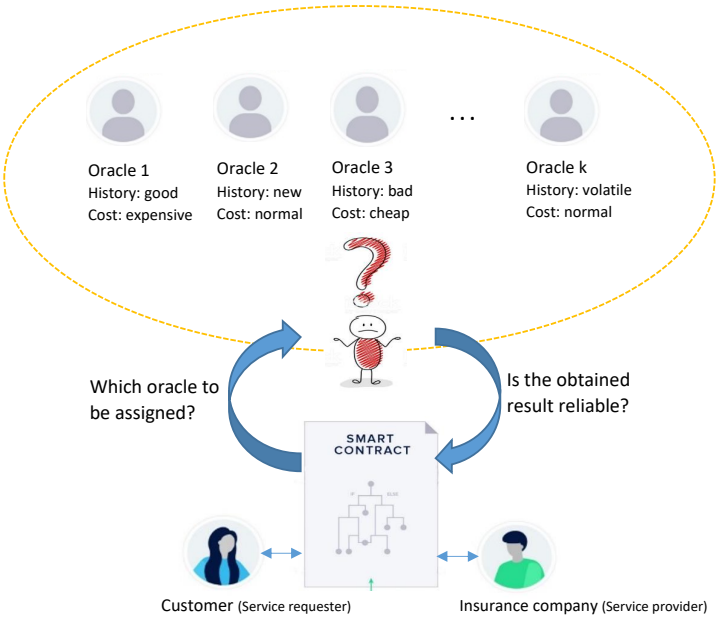


Figure 7.1: Motivational scenario

Figure 7.1 presents the motivating scenario of this paper. Let us assume a smart contract running an insurance marketplace platform in a trustless environment. Imagine that an insured costumer has a car crash and makes a claim to its insurance company. According to the agreed policies signed in the smart contract, the insurance company requires the crash sensors' data and some evidences to process the claim,

but the blockchain network cannot provide such data. To transmit the data and estimate the situation, the smart contract needs to employ an oracle. Since the oracle determines what the smart contract sees, it is crucial to employ not only an economical oracle, but also a reliable and trustworthy one. If the oracle is malicious, it may report in favor of the insurance company and the smart contract accepts the result blindly. Besides, the smart contract cannot rely solely on the historic behavior and reliability of the oracle, since it might decide to deviate for any reason. Furthermore, usually the allocated budget for these tasks are limited. It gets more complicated when some oracles are new with little or no history offering low prices. The smart contract has to find an optimal choice with a balance between reliability and price that all the blockchain validators agree on the obtained result.

Oracles, like human subjects and computer applications, are susceptible to bad behavior that can manifest in gamified attacks in blockchains. In its most basic form, a centralized oracle can supply misleading data, which can impact the actions of blockchain nodes in ways that make them susceptible to attacks. In some situations, the incentives to submit non-truthful data may surpass the gains of acting truthfully. It is argued that a high decentralization of the oracle model would lead to less vulnerability to the "Oracle Paradox" [23]. No matter how centrally centralized it is, an oracle will always come with a price. The most profitable strategy should be always acting honestly, which is why strong incentives must be placed. This raises the need to investigate an incentive mechanism that can motivate oracles to behave honestly. In this study, we argue that assigning a reputation value to these oracles and make them subject to tests against other oracles make the option of reporting misleading data not profitable.

7.3 Related Work

Summaries of the related literature are drawn from two different areas: blockchain and multi-armed bandit.

7.3.1 Blockchain

Blockchain is a distributed database system built upon a timestamped list of transaction records. Its main innovation lies in allowing parties to transact with untrusted parties using a computer network [53]. The blockchain data structure is a hierarchy of blocks that aggregates transactions. Each block is uniquely identifiable and linked to its predecessor in the chain, and integrity is ensured using cryptography-based techniques.

Nodes in a blockchain network might perform arbitrary or malicious behaviors, or possessing misinformation. So, consensus mechanisms are the core of blockchain networks to ensure that all participants agree on the state of the network in such trustless environments [9]. The most important consensus algorithms for blockchains are Proof of Work (PoW) and Proof of Stake (PoS). In blockchains using PoW (e.g. Bitcoin), the algorithm rewards participants for solving cryptography-based puzzles to validate transactions and build new blocks. In PoS-based blockchains (e.g. Ethereum's upcoming Casper implementation), validators take turns to propose and vote for the next block. The weight of validators vote depends on their stake or deposit on the network. PoS provides enhanced scalability, fast transactions, low computation and energy consumption, and high security.

Blockchain Applications and Smart Contracts

Blockchain was originally designed to operate as a trustless peer-to-peer network for financial transactions. Since then, the technology has grown to include many other applications including smart contracts. A smart contract lives on the blockchain and has its own unique address. Moreover, smart contract technology allows users to create autonomous applications that operate independently without any intervention from a system entity. While BLOR can theoretically support any smart contract with low computation surcharge, this paper focuses on its use in Ethereum for implementation due to its publicly accessible platforms. Ethereum initially developed its platform based on PoW, but recently is performing a significant upgrade to presents Ethereum 2.0, using the Casper protocol [11]. The Casper protocol eases the transition from the current PoW to a pure PoS protocol. Ethereum's cryptocurrency is called Ether. In the current version, Ethereum functions through gas which is an Ether-based purchase of the consumed resources. This will help Ethereum prevent DoS attacks, infinite loops within contracts, and in general control network resource expenditure. Every function, such as sending and retrieving data, executing computation, and storing data, has a gas cost.

Smart contracts have two types: deterministic and non-deterministic [57]. The deterministic smart contract code is implemented on a blockchain with complete isolation of external environments, and the decisions and the contract states are maintained by participants within the blockchain. By contrast, the nondeterministic smart contract code needs external information to make decisions, making it dependent on actors outside the blockchain network. For example, the external actor could be a weather information provider or a sensor data provider, who are known as oracles in blockchain.

In recent years, blockchain technology has revolutionized corporate governance

by offering greater transparency among all stakeholders. Easier administration, and the creation of an infrastructure for innovative applications where business transactions could be shared in real-time [96]. A few efforts have been made to study the potential of blockchain in real-world applications despite its great potential for businesses to share data and collaborate in a secure and customized manner [53]. According to Tractica, a market research firm, the annual revenue for enterprise applications of blockchain is estimated to reach $19.9 billion by 2025 [39]. The majority of studies about blockchain's application have focused on finance [85], energy [60] and IoT applications [102].

Blockchain Oracles

The way an oracle retrieves its data depends on whether it relies on human involvement or functions completely automated. Automated oracles operates solely through software and hardware by accessing a data source and retrieving the required data. This means that the oracle itself is fetching the data and is not the original source of the data. Automated oracles only provide deterministic inquiry results as they retrieve existing information from a data source. However, this is not the case with autonomous oracles or oracles involving human intervention. These oracles are not only able to transmit deterministic data, but also to respond to arbitrary inquiries which could be hard to be deducted by machine. Autonomous oracles and human intervention-based Oracles cannot be distinctly separated from the data source. So far, we could not find any paper addressing the issue of trustworthiness for these types of oracles.

Oracle systems can be centralized or decentralized. Oraclize (now is called Provable)[2] is a centralized oracle service based on Amazon Web Service that provides

[2] `https://provable.xyz/`

data feedback for smart contracts and blockchain applications. The main attention of Oraclize is given on proving that the obtained data from its original source is genuine and untampered. Town Crier [99] is also a centralized authenticated data feed that operates as a trusted bridge between existing HTTPS-enabled websites and Ethereum. In fact, it uses trusted hardware and software to be able to prove that the tasks are performed with no tamper and results are reliable. However, similar to any other centralized solution, its validity relies on a central authority and there is no guarantee if the task is performed correctly. It also pays attention to bring data to smart contracts in a trustworthy way, but the data resource is questioned.

Chainlink [23] is a decentralized oracle network on the Ethereum platform. It originally aims to provide tamper-proof data for smart contracts through accessing key data resources using designated APIs. Chainlink operates through incentives and aggregation models, however, it has cost and scalability issues. In another attempt, the authors in [52] proposed a decentralized oracle system equipped with verification and disputation mechanisms. ASTRAEA [6], is an interesting decentralized oracle working based on a voting game to decide about the truthfulness of propositions. All voters place some amount of stake to have the opportunity to vote on a selected randomly proposition. The authors analyzed the game-theoretical incentive structure to prove the existence of Nash equilibrium under the assumption that all rational players behave honestly.

7.3.2 Multi-Armed Bandit

Reinforcement learning is one of the most popular machine learning techniques that is inspired by behavioral psychology of a biological agent. The idea is that an intelligent agent learns the outcome of its actions by interacting with the environment in which it optimizes its actions based on the accumulated rewards it receives. Multi-armed

bandit is a classic reinforcement learning problem. Its name comes from slot machines in casinos, where a gambler is in front of a row of slot machines and he should decide which machine, how many times, and in which order to play the slot machines in order to maximize his potential prize. In this context, each machine gives a stochastic reward from a probability distribution. The gambler's objective is to maximize the total of rewards earned by pulling the sequence of levers at slot machines. Multi-armed bandit problem is exploited in many fields such as medical [86], recommender systems [48], and crowdsourcing [38]. But, its application in blockchain along with its specific challenges has not been explored in any research yet to the best of our knowledge.

There have been very few related academic initiatives utilizing reinforcement learning for blockchain and smart contracts. Among which, one paid major attention in the energy-aware resource management problem in cloud data centers and developed a robust blockchain-based decentralized resource management framework in order to save the energy consumed by the request scheduler [92]. This research utilizes a reinforcement learning model embedded in a smart contract to minimize energy cost. Their simulations based on Google cluster traces and electricity prices showed their method was able to reduce the data centers' cost significantly. Multi-armed bandit is very accurate with minimal complexity and required computing resources comparing to other reinforcement learning methods. These specifications made it a very optimal solution to be employed on blockchain and smart contract platforms where computation and storage are very precious.

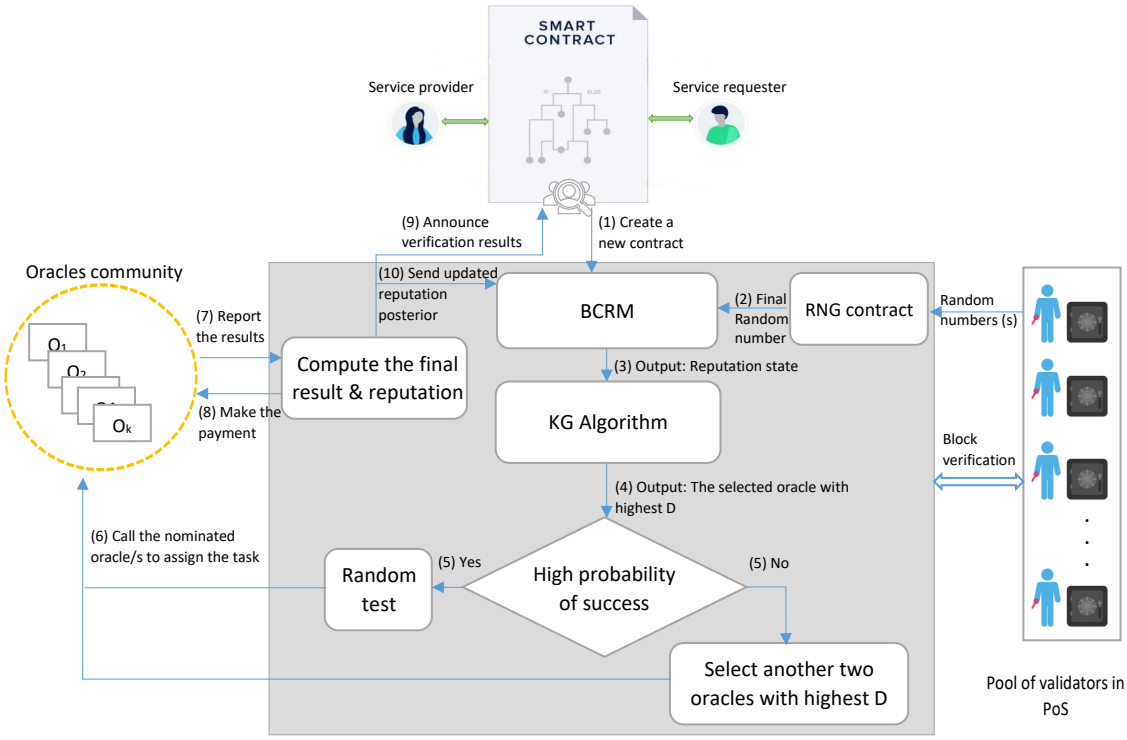


Figure 7.2: BLOR framework

7.4 BLOR: A Markovian Multi-Armed Bandit-based Solution

The main concern of a blockchain-based system, which requires to obtain data from the outside world, is how to maximize total rewards from various oracles in an uncertain setting through trial and observation. BLOR provides an optimal solution using Bayesian theorem and reinforcement learning techniques. In the process of BLOR's sequential decision to choose a proper, reliable and cost efficient oracle, two components have to be considered:

1. Learning: BLOR utilizes observations to update its understanding and knowledge regarding the reliability of oracles.

2. Choice: BLOR selects an action that has a proper balance between the

immediate reward of oracles (short-term objective) and the increased knowledge of oracles (long-term objective).

For the learning component, BLOR uses a novel Bayesian cost-dependent reputation model (BCRM), and for the control component it uses Knowledge Gradient (KG) [26]. BLOR is responsible to manage trust establishment among the blockchain participants and oracles by assigning tasks to the best performing oracles. A digitally assigned reputation value is an effective factor that can be used by BLOR to recognize the premier oracles. To assess and model reputation, we require information and evidence about the history of the evaluated oracle. However, solely employing oracles with good history implies loosing significant opportunities of stranger oracles that BLOR has never encountered before. Furthermore, these oracles tend to be more costly, which can lead to an exceeding budget. Thus, it is crucial to combine the cost factor with the reputation value while assigning a task to an oracle. BCRM assumes that reward rates can change during the experiment depending on the reputation state of the oracle. There are just two possible actions for the choice component: 0 (freeze) which produces no reward nor state change, and 1 (continue) with reward θ_k^t and reputation state changes, according to Markov dynamics.

KG prefers the actions that inspects the choices with little information. Exploration is an endeavor to gain knowledge with a minimal use of precious time and computing resources. Pure exploration can waste time and computing resources if it searches irrelevant areas of the environment. This also means that the agent's learning efficiency may be poor, since it is wasting time on actions that don't contribute to its goals. Finding a good balance between exploration and exploitation is highly beneficial. The agent may be able to discover the most worthy areas to explore by exploiting its current knowledge of the environment. Furthermore, optimizing the cost of learning (i.e., making the agent's performance during learning as high as possible)

cannot be achieved without some level of exploration of the environment, which is important in identifying efficient behaviors.

7.4.1 BLOR Framework

Figure 7.2 illustrates the workflow of BLOR that aims to select the most optimal oracles in terms of reliability and cost. Majority of this process happens on-chain so that the blockchain's validators can verify and achieve a consensus to avoid any bias. Let us consider the insurance company as a service provider and the customer as a service requester, trading through a smart contract. When a claim request is triggered by the service requester through the smart contract, BLOR will decide which oracles will perform the task. At first, once the smart contract receives the request, it automatically creates a new contract with the network of oracles who are the other beneficiary party. This contract contains a new set of rules and conditions such as the payments and compensations policies.

In order to understand the oracles behavior and learn their rewards (in terms of reliability and cost), BLOR creates a BCRM. However, since BCRM is a Bayesian model and probability is involved, we need to generate a random number from the prior distribution in each trial. Generating a random number by each node of the blockchain is a challenge as each node could come up with a different number, around which making a consensus is impossible. Therefore, we propose a Random Number Generator (RNG) with participation of all the validators' nodes. Thus, in step 2, a RNG contract will be created to issue an agreeable random number which will be elaborated in Section 7.4.3. To select the most rewarding oracle, the KG algorithm is used thanks to its high performance and fast computation that makes it suitable for the blockchain environment. In step 3, KG uses the generated reputation state of each oracle to calculate its degree (D).

The KG algorithm selects the most rewarding oracles over a period of time, so at each request, there could be some chosen oracles with unknown or low rewards. For this reason, the chosen oracle has to be checked if additional oracles have to be hired. Depending on the strategies and objectives of the blockchain and its validators, as well as the sensitivity of the task being outsourced to the oracle, the smart contract has to decide if the probability of truthfully and successfully performing the task is high enough for the chosen oracle in step 4. If this probability is low, the next two oracles with highest degree of knowledge (obtained from KG) shall be selected to ensure a reliable result (step 5: No). The intuition behind selecting three in total is to evaluate the trustworthiness of the results based on the majority vote and then rate each oracle accordingly. However, more than three oracles is not economically justified since each of them charges the smart contract to perform its task. If only one oracle is chosen, we can have some random tests by using other oracles from time to time to use for training our model (step 5: Yes). These random tests ensure that even trusted oracles do not behave maliciously and the learning is processed without deviation. The nominated oracles will be called by smart contract triggers within step 6 and the results will be reported by sending a signed transaction to BLOR in step 7. BLOR verifies the results, updates the reputation values of the participated oracles, then pays these oracles according to the defined rules and conditions in step 8, and informs the requester about the result (step 9). In the last step (10), the updated posterior reputation including the success or failure of the oracle will be sent to BCRM.

7.4.2 Formulating Oracles Problem in a Bandit Setting

Suppose on each data request, we have K oracles known with reward rates, θ_k, $k = 1, \ldots, K$. At first, we assume θ_k to be the true reward mean if oracle k is to be chosen.

We do not know θ_k, but we assume that it is normally distributed with prior mean θ_k^0, variance $(\sigma_k^0)^2$ and precision $\beta_k^0 = 1/(\sigma_k^0)^2$. Let $R^t = (\theta_k^t, \beta_k^t)$ be the vector of reputation states with the means and precisions for all the choices of oracles after t trials.

Let k^t be the oracle that we choose after t trials, meaning that our first choice is k^0 made based on the prior, purely. These trials are made based on a policy π to be run by smart contract which depends on the history of trials. Policy is a decision rule that BLOR adopts on behalf of all the blockchain participant to assign tasks to oracles. Let us assume that K_k^π is the random variable representing the total number of selecting oracle k, given the policy π. This number is random since the results depend on the observed rewards. Our objective is to choose a policy π that solves the following supremum objective function $supV$ where R_k is the reputation state of the oracle k:

$$supV^\pi = \mathbb{E}^\pi \sum_{k=1}^{K} K_k^\pi R_k \theta_k \tag{7.1}$$

$\mathbb{E}^\pi$ stands for the expected value depending on π to reflect the underlying probability space that we are going to construct. Learning problems can be easily formulated in a Bayesian framework, where we are able to capture the uncertainty in our belief about a system. In our oracle bandit problem, θ_k is the true rewards value of oracle k, but we do not know this value. Instead, we assign a probability distribution from the Beta distribution that describes what we think θ_k is for each oracle. Since each oracle can have two outcomes of success or failure, we employ the Beta distribution where trials are generated independently and identically from an unknown Bernoulli distribution for each oracle. The following section explains how we construct our Bayesian model for oracles.

7.4.3 Bayesian Cost-dependent Reputation Model

We formulate two components of Bayesian learning as follow: 1) Bayesian Inference: to update the reputation representing the belief about the probability of a successful and truthful evaluation (reward) based on new information; and 2) Bayesian Learning: to compute the posterior probability distribution of the target features.

Usually bandit solutions using Bayesian learning assume that there is a fixed probability of $\gamma \in [0, 1]$ for the experiment repeated on any given trial t. Then, the appropriate value of γ shall be learned over the time period of experiment. This approach is naive to solve the problem of oracles, since oracles might change their behavior and deviate in a certain point of time for certain cases. For this reason, we adopt a dynamic model for reputation state in which γ^t has a Markovian dependency on γ^{t-1}. A graphical illustration of these two models is presented in Figure 7.3.

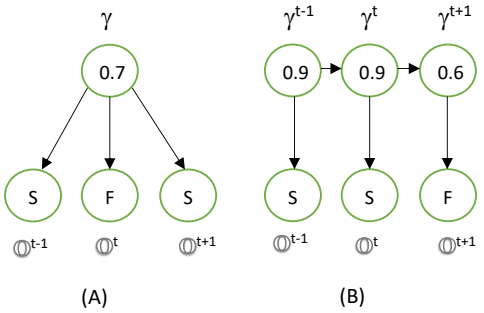


Figure 7.3: Graphical model of (A) Fixed model vs. (B) Dynamic model. The numbers in circles show example values for the variables. S denotes the oracle success and F denotes the oracle failure to deliver reliable results with a fixed probability of γ or dynamic probability of γ^t.

Under the dynamic reputation model, the reward probabilities might change at times during the experimental session, as each oracle is an autonomous agent in our problem and might change its behavior. Thus, during any trial, the prior reputation of each oracle is a combination of the posterior reputation from the previous trial and a generic prior. The main task of BLOR would be to track the evolving reward

probability of each oracle during the period of trials. We assume the prior distribution that generates the Bernoulli rates is a Beta distribution, Beta (a, b), which is conjugate to the Bernoulli distribution, and whose two hyper-parameters, a and b, specify the pseudo-counts associated with the prior.

Random Number Generator

Let n be the number of validators. In BLOR, the random number is generated by all the n validators to enable the final verification of results. First, we need to create a Random Number Generator (RNG) contract in BLOR, which defines the participation rules and computes the final random number. The basic process of generating a random number can be divided into two phases:

1. Any validator who wants to participate in the random number generation and the final candidate verification needs to send a secret number ($s_i \in \text{Beta}(a, b), 1 \leq i \leq n$) encrypted by Keccak-256 hash algorithm through a transaction to the RNG contract in a specified time period (e.g, 3 blocks period). The RNG contract will check if s is valid by running Keccak-256 against s. Valid s is kept to calculate the final random number.

2. After collecting all the secret numbers, the RNG contract calculates the random number from the function $f(s)$ and the final random number will be sent to BLOR and all the validators, where s_{min} and s_{max} are the minimum and maximum numbers respectively:

$$f(s) = \frac{\sum_{i=1}^{n} s_i - s_{min}}{s_{max} - s_{min}} \tag{7.2}$$

Oracles Reputation Model

Let S_k^t and F_k^t be the numbers of successes and failures obtained from the kth oracle after t trials, and θ_k^t to be the estimated reward probability of oracle k at trial t. θ_k^t has a Markovian dependency on θ_k^{t-1}, so that with probability γ, $\theta_k^t = \theta_k^{t-1}$. Also with probability $1-\gamma$, θ_k^t is redrawn from the prior distribution Beta(a, b). To infer the new posterior distributions, BLOR combines the sequentially generated prior reputations with incoming observations (successes and failures on each oracle). The observation $\mathbb{O}_k^t$ is assumed to be Bernoulli(θ_k^t).

In order to maximize the utility of the blockchain's participants, BLOR needs to select the oracles that perform their tasks correctly at a lower price. Let c_t be the normalized cost that oracle k charges the blockchain with a weight of w to adjust the value of cost to the chain participants. We denote $R(\theta_k^t)$ as reputation state which is the posterior distribution of θ_k^t given the observed sequence. At each trial, the updated cost-based reputation state can be computed using Bayes' rule as follows:

$$R(\theta_k^t) \sim P(\mathbb{O}_k^t \mid \theta_k^t) P(\theta_k^t \mid S_k^{t-1}, F_k^{t-1}) / w c_t \tag{7.3}$$

where, the prior probability of reward state is a weighted sum (parameterized by γ) of last trial's posterior and the generic prior $R^0 \equiv f(s)$, as defined bellow:

$$P(\theta_k^t = \theta \mid S_k^{t-1}, F_k^{t-1}) = \gamma R_k^{t-1}(\theta) + (1-\gamma) R^0(\theta) \tag{7.4}$$

7.4.4 BLOR's Final Decision based on Knowledge Gradient

In reinforcement learning methods, the entire reward is generally received after the final measurement, which is impractical for our problem in blockchain. Thus, we need to receive the reward given in pieces over time. This will not only decrease

the complexity of the solution and computational resources, but also will shorten the results time to identify the best oracles from the beginning of the process in an online manner. The KG policy can achieve this by maximizing the single period reward.

The objective given by Eq.7.1 and Eq.7.3 has a terminal reward, $V^{T,\pi}(R^T) := \max \theta_k^t$. However, we require to restructure it in order to provide single period reward $V^{T,\pi}(R^t)$ at trial t, and $V^{T,\pi}(R^{t+1}) - V^{T,\pi}(R^t)$ at times $t+1, \ldots, T$, meaning that:

$$\begin{aligned}\max \theta_k^t =& [V^{T,\pi}(R^T) - V^{T,\pi}(R^{T-1})] + \cdots + \\ & [V^{T,\pi}(R^{t+1}) - V^{T,\pi}(R^t)] + V^{T,\pi}(R^t)\end{aligned}$$

KG is defined as a single-step and look-ahead policy, which selects the next instance with the largest expected reward, greedily. Its algorithm is close to the optimal policy. While pretending only one more exploratory measurement is allowed, it assumes that after the next measurement all remaining choices will exploit what is known. It evaluates the expected change in each estimated reward rate for each oracle, according to the current reputation state R_k^t. The approximate value function for choosing oracle k, $D^t = k$, on trial t is:

$$V_k^t = \mathbb{E}[\max \theta_k^{t+1} \mid D^t = k, R^t] - \max \theta_k^t \tag{7.5}$$

The first term is the expected largest reward rate (the value of the subsequent exploitative choices) on the next step if the kth oracle were to be chosen, with the expectation taken over all possible outcomes of choosing k. The second term is the expected largest reward given no more exploitative choices. Their difference is the "knowledge gradient" of taking one more exploratory sample.

Let us imagine we have T trials of which $(t-1)$ measurements were already

made. For the tth measurement, the KG decision rule is defined as follows:

$$D^t = \arg\max \theta_k^t + (T - t - 1)V_k^t \tag{7.6}$$

Other than very minimal computational resources that validators require to compute KG, it accommodates the issue of cold start for oracles who join the network later on. It means that the KG policy measures those alternatives that has less knowledge about. The predictive distributions of these alternative oracles (k') have large variance $(\sigma_{k'}^t)^2 > (\sigma_k^t)^2$, or equivalently, small precision $\beta_{k'}^t < \beta_k^t$.

7.4.5 An Illustrative Example

In this section, we provide an illustrative example to show how BLOR works in details. At first, BLOR shall construct BCRM. Assume that we have 5 oracles, $K = 5$, $O1, \ldots, O5$. For each one of them, we shall calculate the reputation state. Figure 7.4 illustrates how we form that for each oracle. Consider $O1$ is measured for the first trial and returns a success. In the beginning, we have no knowledge of its performance, therefore its prior for the next step ($S1$) is equally distributed in the Beta setting. Since computation of prior includes randomness, according to Figure 7.2, BLOR calls the node in charge of generating random numbers to have the same random number for all the validators. The summation of the prior with the random number is multiplied by the Bernoulli trial to obtain the posterior. This posterior will be used as the prior for the next step ($S2$). During the next trial ($S3$), $O1$ returns a failure which negatively affects the posterior distribution.

After creating BCRM, BLOR seeks to find out the best oracles for the task by computing and comparing their KG degrees. Table 7.1 presents these calculations. Let us assume that the total number of trials is 500 and BLOR is

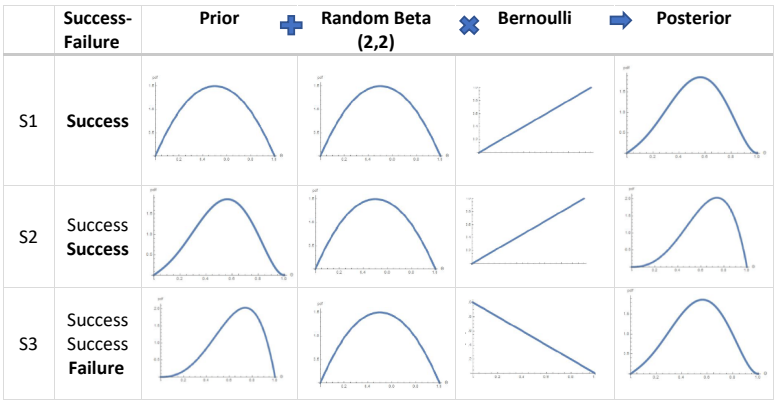


Figure 7.4: Illustration of partial rewards state of $O1$ within the constructed BCRM

exploring its 65th trial, $(T = 500, t = 65)$. We further assume that at $t = 64$, $O1$ is selected and its number of successes became 20. BLOR must compute the expected reward rate, value function and KG decision using Eq.7.3, Eq.7.5, and Eq.7.6. Consider the two oracles $O1$ and $O4$, which have the highest numbers of successes according to Table 7.1, with a reputation state of 0.66 obtained from the previous trial.

1- Choosing $O1$:

$$\begin{cases} O1 \ wins \xrightarrow{(P=0.67)} & R_1^{65} = 0.68/0.8 = 0.85 \\ O1 \ loses \xrightarrow{(P=0.33)} & R_1^{65} = 0.65/0.8 = 0.81 \end{cases}$$

Therefore, we have:

$V_1^{65} = (0.67 * 0.85 + 0.33 * 0.81) - 0.66 = 0.16$

$D^{65} = 0.85 + (436 * 0.16) = 70.61$

1- Choosing $O4$:

$$\begin{cases} O4\ wins \xrightarrow{(P=0.65)} & R_4^{65} = 0.68/0.7 = 0.97 \\ O4\ loses \xrightarrow{(P=0.35)} & R_4^{65} = 0.66/0.7 = 0.94 \end{cases}$$

Therefore, we have:

$V_4^{65} = (0.65 * 0.97 + 0.35 * 0.94) - 0.66 = 0.95$

$D^{65} = 0.97 + (436 * 0.95) = 415.17$

Thus, among $O1$ and $O4$, $O4$ with the highest degree of KG will be selected. Please note that to save computation time, BLOR does not calculate value function and KG for the oracles with very low chance of being selected.

Table 7.1: BLOR in the illustrative example ($t = 65$)

$T = 500$	$O1$	$O2$	$O3$	$O4$	$O5$
Success	20	5	0	10	5
Failure	5	10	5	0	5
Cost	0.8	0.5	0.5	0.7	0.6
$R(\theta_k^t)$	0.85	0.82	0.7	0.97	0.83
V_k^t	0.16	0	0	0.95	0

The algorithm of BLOR will be explained in the next section in a case study where we construct smart contracts and the relationship among them.

7.5 A Case Study of Cloudchain (Cloud Services Trading over Blockchain)

The aim of the Cloudchain case study is to present how BLOR can offer a unique smart model for employment of oracles and transform the way cloud services are delivered. Cloudchain [82] is an innovative distributed blockchain-based framework to support interoperability and coopetition (i.e. cooperative competition) among the

cloud providers. Cloudchain allows the cloud providers to outsource their unmet computing demands.

Utilizing smart contracts in blockchain enabled Cloudchain to offer higher transparency, visibility, and reliance within its fully decentralized agreements deployed on top of Ethereum. However, Cloudchain falls short in supervising the SLA's agreed terms, which requires to access the outside world of the blockhcain network. Each of the cloud providers may disagree about the SLA compliance. Yet, investigating that is beyond the control of blockchain validators or digital codes embedded in the smart contracts due to its self-contained execution environment. Thus, oracle is required to perform the highly important verification task and confirm if the SLA is met. In many research works, oracle is assumed to be a fully-trusted third-party agent that has access to the outside world, and feeds the data into the blockchain to be accessible by the applications [83]. Furthermore, oracle is assumed to be single or act as a member of a group, while in real world, each oracle could be a selfish agent trying to maximize its own gain. We explain how BLOR can contribute to this situation.

7.5.1 Background: Cloudchain's Smart Contracts

Three types of smart contracts are incorporated into Cloudchain that include executable functions and state variables. [82].

Contract 1 or Cloudchain Registry (CCR) is a global contract mapping Ethereum addresses (equivalent to public keys) to cloud providers identification values (including *Name*, *Reputation Value*, *Computing Capacity* and *Storage Capacity*). A contract can include policies governing the registration of new providers or changes to the mappings of existing ones. Only certified cloud providers can register for the cloud provider program. In addition, CCR maps identities to Cloudchain Contract (CCC) addresses, where an exclusive contract concerning each provider profile and

list of services is recorded.

Contract 2 provides Cloudchain Profile (CCP). It contains a list of references to CCC, which represent all the participants' prior and current dealings with other nodes in the system. CCP also implements a feature that enables provider notifications. Ethereum supports the creation of events to indicate that certain actions have been performed (e.g. an update to profile's data). Providers must submit their requests to the CCP contract to be propagated and raised to other nodes. Each transaction records a status variable. This indicates if a transaction is newly initiated, waiting for updates, or has been completed. This contract is critical since it stores new CCC contract addresses, and without it Cloudchain might lose track of all the contracts.

Contract 3 indicates the Cloudchain Contract (CCC). It is created between two nodes when one agrees to provide the requested service to the other. Similarly, the contract can be completed or canceled by the beneficiaries. The contract balance would be transferred once the contract is completed or canceled, and the status of the contract would also be updated. Cloudchain members are able to join and leave the system at any time by executing functions in smart contracts. These flexible memberships allow members to supply or demand services once or multiple times as required.

7.5.2 BLOR in Cloudchain

Figure 7.5 illustrates interactions among the contracts and cloud providers. In step 1, The Cloud Provider as a requester (CP_r) and the Cloud Provider as a supplier (CP_s) should register in CCR. Public key pairs are assigned to each registered user in CCR.

In the case of a computational resource deficiency, CP_r can create a new CCC in step 2 by requesting a service using CCP. CCP ties identities to the CCC's address on the blockchain and keeps a history of providers previous and current engagements

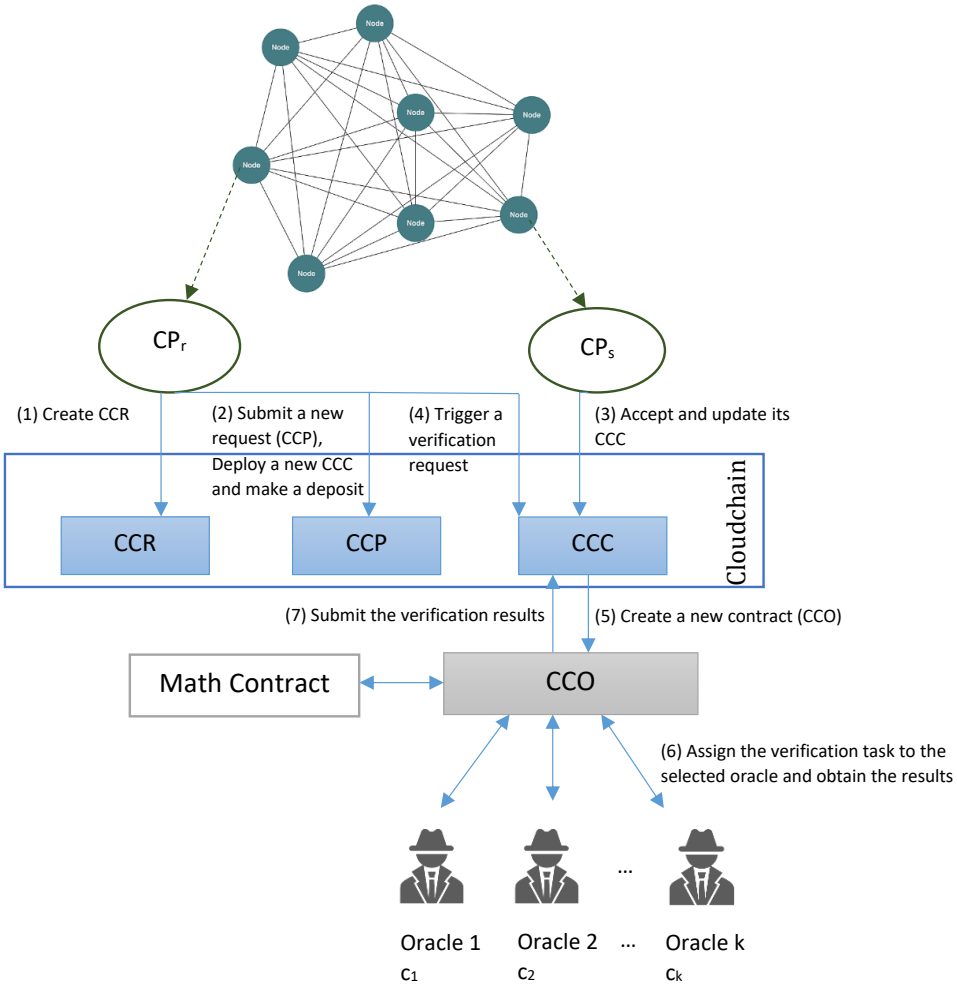


Figure 7.5: Application of BLOR within Cloudchain

with other nodes in the system as well as any SLA violations.

CCC allows the interaction between two nodes in the network in which one node responds to the other's request in step 3. In order to complete a contract, CP_r is required to pay a deposit in advance. Beneficiaries can choose to end the contract or cancel it. However, the contract termination and delivery of the requested service must be confirmed by CP_r. Once the contract is complete or canceled, CCC will calculate and charge fines if any exist. The balance remaining on the contract would be transferred to the CP_r or CP_s address accordingly. The status of the contract would be updated as well.

In step 4, CP_r can initiate quality monitoring at any time to verify whether the provider meets the SLA conditions during the runtime. To do so, the request should be submitted to CCC in which a new contract of CloudChain Oracle (CCO) will be created to perform the verification in step 5. CCO holds a list of all oracles and their past history. Each oracle announces its cost to perform the task c_k. CCO implements BLOR to find qualified oracle/s to monitor the SLA. CCO runs the designed models and formulas through a set of defined functions. Since the current blockchain platforms are not capable of implementing complex math functions, a math contract is defined to perform basic operations in order to calculate advanced functions. Once one or multiple oracles are nominated, the verification task is assigned and the proper money is deposited into CCO in step 6. In the last step (6), the obtained result is extracted to be push into CCC (step 7) and be used to train the defined models in BLOR.

Function calls on contracts involve transactions, and those which make changes to the contract storage must be validated by blockchain validators. Once a block is mined with the newly linked contract, it will be broadcast to other nodes, and the first node to accept the request will update the contract accordingly.

All the explained procedure and interactions among smart contracts are elaborated in Algorithm 7.1 and 7.2. Algorithm 7.1 illustrates the process of requesting a service and triggering a quality monitoring request and Algorithm 7.2 presents the process of selecting the best oracle/s by BLOR.

7.6 Experimental Results

Because there is no available dataset about blockchains' oracles, in order to evaluate the performance of BLOR, we simulated 100 oracles operating within a blockchain in

Algorithm 7.1 Cloud providers service agreements within Cloudchain

Require: Ether deposit; Cloud requester's Etheruem address (CP_r); Cloud supplier's Etheruem address (CP_s); c_k; CCO Ethereum address.

procedure ServiceAgreement
    CP_r makes a service request in CCP
    CCP creates a CCC
    CP_r.SendTo(CCC, Ether deposit)
    CCC.Availability = True
    EventLog.Create("New request is available")
    while CP_r requests a quality verification **do**
        CCC calls BLOR in ***Algorithm 7.2*** ▷ Outsource the task to oracle/s to obtain the verification result
        CCC.SendTo(CCO, c_k) ▷ Pay the cost of the oracle c_k to perform the task
    end while
    if CCC.Completed **then**
        EventLog.Create("CCC is completed")
        ContractDeposit = CCC.TotalAmount
        CCC.SendTo(CP_s, ContractDeposit)
        EventLog.Create("Fund is transferred to the Cloud supplier")
    end if
end procedure

1000 observations. We implemented and experimented with BLOR using Python on Google Colab and the Solidity language on Ethereum. Because a bandit is an online learner, it needs a record of the oracles history prior to the current time step we are simulating in order for it to act like it will in a production setting. Each oracle is assumed to have a different historical performance drawn from a beta distribution. The normalized cost of each oracle is assumed to be fixed and normally distributed with a mean of 0.54 and standard deviation of 0.17. We first discuss the challenges that we dealt with, and then provide the obtained results.

7.6.1 Implementation Issues

The current version of Cloudchain is using Ethereum and it is not easy to run a machine learning algorithm on Ethereum that is a public blockchain with various limitations. Besides, at the time of writing this research study, Ethereum is still using PoW consensus and has not been upgraded to PoS yet. We came up with some

Algorithm 7.2 BLOR process within CCO

Require: CCC Ethereum address; Oracles' Etheruem address; c_k; Ether deposit.
Ensure: Verification.Result ▷ Boolean

procedure OracleSelection
    CCC.Deposit(CCO, Ether deposit)
    Retrieve CCC's terms and conditions to monitor the service
    RNGContract.getRandom()
    MathContract.calculate($R(\theta_k^t)$) ▷ refer to Eqs. 7.3, 7.4
    BLOR calculates D^t and selects the oracle with the highest D^t ▷ refer to Eqs. 7.5, 7.6
    if probability of success is low **then**
        Select another two oracles with highest D^t
        if EachVerification.Result = True for majority of oracles **then**
            Verification.Result = True
        else if EachVerification.Result = False for majority of oracles **then**
            Verification.Result = False
        end if
    else
        Test randomly against oracles with high reputations
    end if
    CCO.SendTo(selected oracle/s, ContractDeposit)
    CCO.Update(Oracle/s reputation/s, Results)
    CCO.ReportTo(CCC contract, Verification.Results)
    Math contract receives the results and updates the posterior reputation of the selected oracle/s in BCRM
end procedure

solutions to test BLOR on the current platform of Ethereum.

- **Random number:** In Blockchain, there is no pure random generator mechanism, because when the code is being run by other nodes, they all should reach the same result to achieve a consensus. There are some possible scenarios to generate a random number such as using a centralized system using an oracle, publicly verifiable secret sharing, or even hash-block. For the purpose of our simulation, we used a simple, yet efficient solution, which is using the block number to generate a hash number to be employed as a random number. This solution is practical and efficient since the block number is not known before being generated.

- **Float number:** Blockchain does not support any float/decimal number with floating points. The reason is because all CPUs work based on a binary

mechanism, and there is no exact representation of fractions in binary mode, so they are round to the nearest match. For this very important reason, Blockchains do not support any number with floating point. Even for financial transactions, they have introduced smaller units such as wei, gwei, etc. instead of using float numbers. Basically, the only supported numerical data type in Ethereum is Integer (either signed or unsigned). So, we require to scale up all the variables in integer level. For instance, if we want to take a number between 0 and 1, we have to change the scale to 0 and 100 to replicate the behavior of $0-1$ with one or two floating points precision. This would also impact parts of the algorithm, since the mathematical behavior of $0-1$ is different from $1-100$. So, the formulas need slight adjustments.

- **Limited number of variables:** In Solidity, Ethereum language in which Cloudchain is coded, there is a limitation for the number of variables which can be initialized and used in a function. When there are too many variables, Ethereum virtual machine does not compile the contracts. We designed carefully our functions to avoid such a problem.

- **Advanced mathematical functions:** Blockchain languages do not support complex mathematical functions by default due to various issues such as the ones discussed earlier. In the BLOR algorithm, we used complex mathematical functions. Thus, in order to run BLOR on Blockchain, we built a new contract called Math Contract. Math contract implements our required functions using four primitive operations only. This contract supports Sin, Cos, Log, exponential, Gamma function, SQRT, Beta Distribution etc. All these functions are developed in Solidity solely based on integer numbers (with no floating point) and primitive operations.

Table 7.2: Probability of selection for the sample oracles

Oracle ID	0	2	7	27	85	95
Cost	0.3	0.7	0.9	0.6	0.5	0.5
Success	4	5	9	6	3	1
Failure	5	0	1	3	4	3

7.6.2 Benchmarking and Simulation Results

In order to assess the effect of performance and cost in BLOR decision making, we compared six oracles containing half faulty, during first 100 observations. Table 7.2 represents their detailed histories and costs. Figure 7.6 presents the variation of value function for each oracle. At the beginning, Oracles with shorter history, such as Oracle 95 and Oracle 2, provide more value in the learning process of BLOR. Consequently, they have a higher chance to be selected by BLOR, as can be seen in Figure 7.7. After few observations, more value is earned with the oracles with cheaper price (such as Oracle 0 and Oracle 85). However, the chance to be selected is more among the oracles with a balance of price and performance (Oracle 27) and those that are very cheap (Oracle 0). The value function of all the oracles tend to zero after a while, when BLOR learned their behavior and there is no more value in exploring them. Meanwhile, Oracle 85 generates an unsteady value, which means BLOR is willing to measure the change into the future expected reward of this oracle. This could be because of gaining more successful history combined with its good price.

Our multi-armed bandit-based solution for the oracle selection problem can vary based on how we do exploration and exploitation. We compare the performance of BLOR against other algorithms as follows:

- No exploitation: this is the most naive approach where the system selects randomly.
- Exploitation with exploration at random: ϵ-greedy is among the most popular

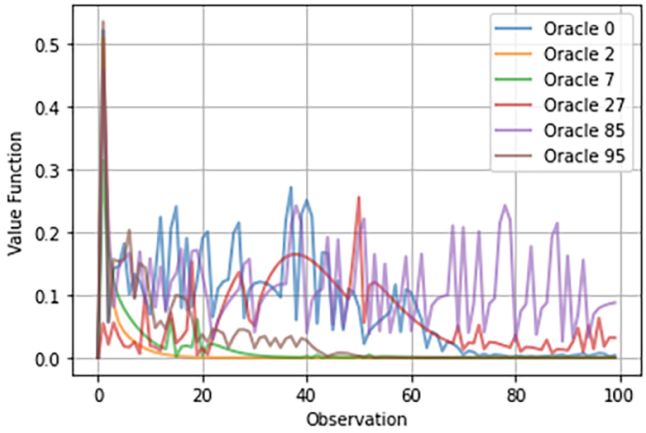


Figure 7.6: Cost and history of the sample oracles

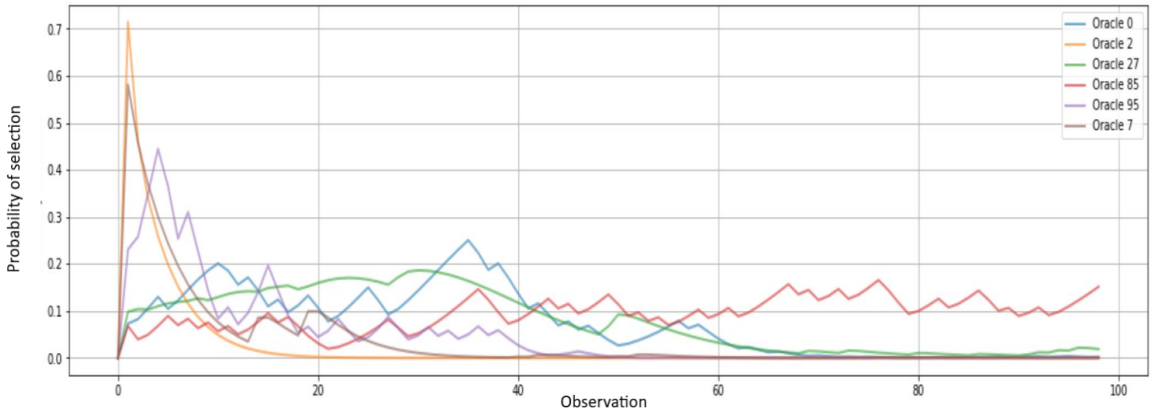


Figure 7.7: Comparison of value function for the sample oracles

and efficient methods of this group. The ϵ-greedy is a heuristic model that assumes decision-making is determined by a parameter ϵ to control the balance between random exploration and exploitation. With probability of ϵ, the oracle is chosen randomly, and $(1 - \epsilon)$ the oracle with the greatest estimated reward rate will be selected.

- Exploration smartly with preference to uncertainty: BLOR is in this category.

We further investigate the performance of Markov chain Monte Carlo in our problem. Markov chain Monte Carlo is a probabilistic machine learning method that creates samples from a continuous random variable. The experiments are conducted using

Table 7.3: Algorithms comparison

	Performance	Cost	Time (mean; STD)
BLOR	90	52	(0.04; 0.005)
Monte Carlo	74	58	(10.3; 1.07)
Random	62	54	(0.008; 0.0003)
ϵ-greedy	82	66	(0.03; 0.003)

Python in Google Colab with Intel(R) Xeon(R) CPU @ 2.00GHz and 13GB RAM. Table 7.3 presents the average performance, cost and elapsed time of each algorithm with mean and standard deviation (STD). Here, cost is the total money that has to be paid to the selected oracles within all the observations. In general, BLOR had the highest performance and Random the least. From the economic perspective, BLOR was the most economical solution and ϵ-greedy was the costliest one. However, in terms of computation time, Random runs very fast, followed by ϵ-greedy with a comparable time against BLOR. As expected, Monte Carlo lasted the longest.

Figure 7.8 presents the performance of each method in noisy observations. By noise, we mean that the oracle did not behave as expected. It is obvious that the performance of all the methods decline when the noise increases, however, BLOR had the most steady accuracy. Even in a very noisy situation, BLOR could maintain its performance by almost 80%. This means that out of the all oracles picked by BLOR, 80% of them could report a successful result. After BLOR, even though ϵ-greedy had higher accuracy, it was the most influenced by the noise. Since this heuristic algorithm mostly picks the oracle with highest reward, it is unable to recognize its change of behavior and is not suitable for noisy subjects. As expected, random selection did not show a significant change in its performance, which was mostly around 50%. The moderate performance of Markov chain Monte Carlo was mainly because of two reasons: 1) this method requires several observations to build its model; 2) Markov Chain Monte Carlo needs equal records of historical data for all oracles, while we

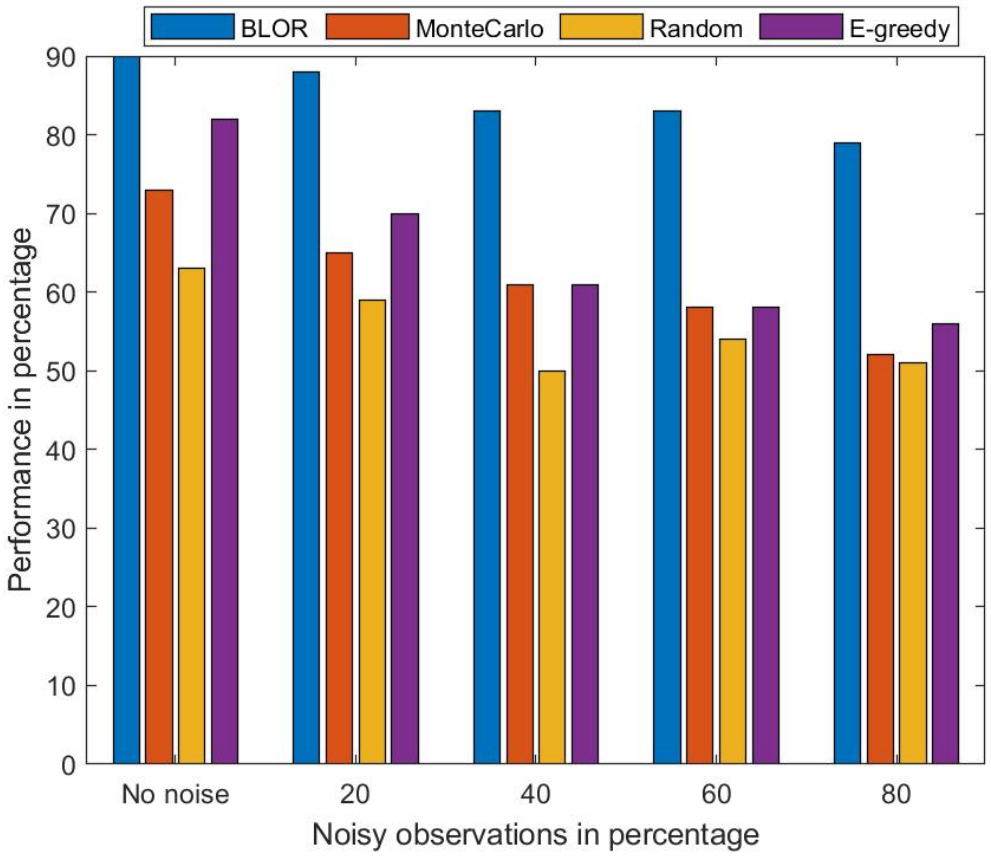


Figure 7.8: Performance comparison against noisy observations

assumed each oracle joined the system at different time.

We further measured the total cost of the selected oracles by all the considered methods, indicated in Figure 7.9. In a non-noisy or less noisy environment, BLOR performed very well. However, as the noise increases, BLOR sacrifices the cost to maintain the high performance. In a very noisy situation (more than 50%), BLOR is the costliest method. ϵ-greedy picked the most expensive oracles in non-noisy or less noisy situations and followed the same expense as the noise increased. Random and then Markov Chain Monte Carlo were economical in all the situations.

In order to investigate the effect of number of faulty oracles on the performance of these methods, we run several experiments with different percentages of faulty oracles. A faulty oracle is an oracle with a history of more than 50% failures. As

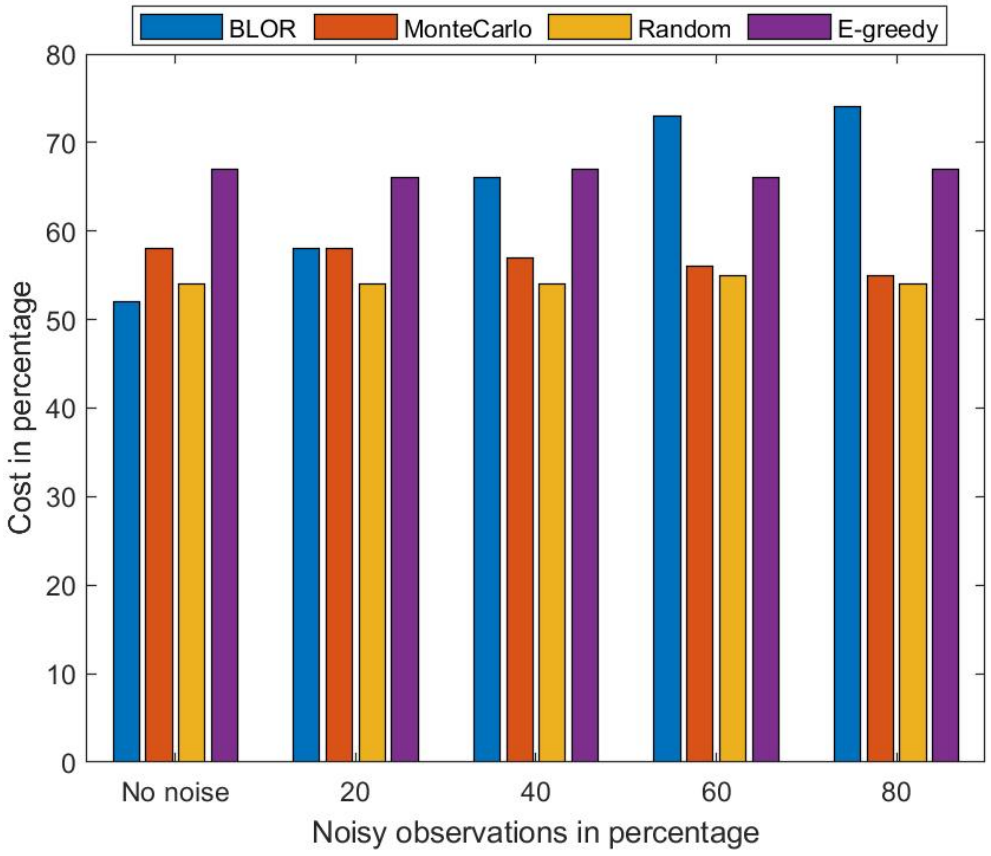


Figure 7.9: Total cost comparison against noisy observations

Figure 7.10 shows, performance or accuracy of BLOR was the highest among these methods, followed by ϵ-greedy. However, as the percentage of faulty oracles decreases, the performance of all the methods increases.

7.7 Conclusion

Oracles gather information from the real world and transport it onto the blockchain for further use. Hence, the use of oracles is imperative to promote a widespread adoption of smart contracts. Yet, research about oracles and their practical application is very immature. As our last contribution, this research study tried to shed some light by addressing two major challenges in this area. The first challenge is about employing

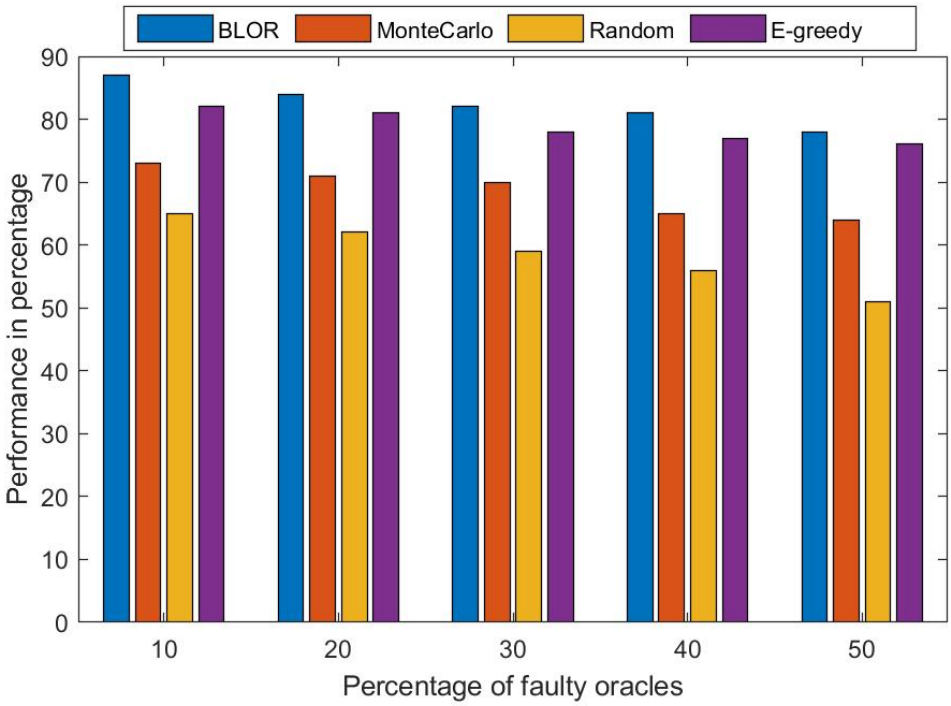


Figure 7.10: Performance comparison against the number of faulty oracles

a smart mechanism in place to identify the trustless and cost-efficient oracles. This challenge was addressed by developing a Bayesian cost-dependent reputation model in a multi-armed bandit setting, named BLOR. The second challenge of actual application of a smart mechanism using reinforcement learning was dealt with by implementing BLOR on Ethereum. To the best of our knowledge, this study is the first to implement a machine learning algorithm for smart contracts in general, and for oracles' recruitment in particular. We showed how to solve various challenges that could raise while implementing complex algorithms like machine learning in Ethereum. To prove the efficiency of BLOR, we simulated 100 oracles operating within a blockchain in 1000 observations and benchmarked it against several algorithms vary by their degree of exploration and exploitation. It was found that BLOR prioritizes the newer oracles which hold less history and those with a fair balance of performance

and price. Through a benchmarking experiment, BLOR proved a steady performance in selecting the successful oracles within a low to high noisy environment ranging, in average, from 80 to 90 percent out of the all selected oracles, whereas, other algorithms performance was ranged, in average, from 50 to 80 percent. BLOR had the most cost saving selection when the noise was lower. In overall, BLOR performed competitively better than the other algorithms, but at the cost of time.

Chapter 8

Conclusion

8.1 Summary and Discussion

In this thesis, we discussed the problems to be tackled and objectives to be achieved within the cloud computing market area. This research has been carried out from two strategic viewpoints of competition and cooperation in order to contribute to the current and future cloud computing market. From the competition perspective, the focus was to enhance the users satisfaction and optimize the profit of service providers who operate within an on-line rating platform by designing and solving a Stackelberg game model between a typical cloud provider and the users. The theoretical results we obtained were confirmed by the game simulation on a real world dataset and showed that rating improvement is the best strategy for high rated providers who offer quality competitive services. This strategy allows those providers to attract the users who prioritize quality in their decision making. Providers with higher capacity, rating and also cost can make more profit when the user demands increase. Providers with lower capacity, cost and rating may see some unexpected increase of demand from some customers, but in total, they will attract less demand and make less profit. Yet

their main advantage is lower cost that attracts low budget customers with continuing their price reduction. These findings addressed the first research question about how to model the conflicting interests of the cloud service providers and consumers.

After identifying cloud consumers' interests and revealing the issues surrounding low rated cloud providers in our first research, we included market competition in our model and introduced a game theoretical framework to allow new and small cloud providers to obtain a market share using their own strategic advantages. The conducted experiments using real-world dataset showed that the user demand from small cloud providers increases the most when these providers offer added-value services. Regardless of the pricing strategy, improving the quality and ratings of a small and new IaaS provider increases its demands' rate and profit. However, the best strategy for small cloud providers is to set higher price and improve the quality of their provided added-value solutions, specifically in the early stages of development. These findings answered our second research question, which is about enabling a productive cloud market industry that includes new and small providers.

From the cooperation perspective, we introduced a new distributed blockchain-based framework, named Cloudchain, for cloud providers federation to overcome the limitations of conventional centralized federations. Due to the coopetitive environment of Cloudchain, and high expense of public smart contracts, we further designed and solved a differential game. This game modeled the best strategies of cloud providers to make a request with an optimal transaction cost and time and optimize their reputation value to receive the requests from other providers. Cloudchain was implemented using Solidity over the Ethereum network and the differential game was simulated for a sample of five cloud providers during 100 days. It was found that for cloud requesters with a high number of requests, spending high gas price is not economically appealing. The results showed that cloud suppliers

have minimal gas consumption, which makes it more affordable for them to pay higher prices and enhance their communication and reputation. Though increasing the reputation was not always the best strategy for highly reputed cloud providers, a gradual increase is recommended. These results address our third research question regarding how to design a blockchain-based cloud federation and how to maximize the providers profit within the platform.

To overcome the issue of compromised QoS within Cloudchain, we proposed a multi-agent blockchain-based quality monitoring model that contains an oracle playing the role of a verifier agent to evaluate the service quality. A Stackelberg differential game was also designed to formulate the best strategies of resource provisioning, the required number of quality verification requests and the monitoring price for the provider, requester and verifier agents, respectively. The evaluation results showed that at the beginning, the provider has to preserve the desired amount of capacity to satisfy the required quality even throughout the peak-times. However, it is not economically justified to make this reservation for the last periods of its contracts' time. Such a resource provisioning impacted the verifier pricing strategy and the number of requests. The reputation of the provider agent elevated the profit of the requester agent. Furthermore, higher penalty raised the capacity and reduced the number of verification requests at the equilibrium. The developed system was proven to be economical for cloud beneficiaries, valuable in transparency and efficient in preventing the SLA violation.

In the last contribution, we shed some light on two major challenges related to oracles and their practical application. The first challenge is about employing a smart mechanism to identify the trustless and cost-efficient oracles. This challenge was addressed by developing a Bayesian cost-dependent reputation model in a multi-armed bandit setting, named BLOR. The second challenge related to the actual application

of a smart mechanism using reinforcement learning was dealt with by implementing BLOR on Ethereum. To the best of our knowledge, this study is the first to implement a machine learning algorithm for smart contracts in general, and for oracles' recruitment in particular. We showed how to solve various challenges that could raise while implementing our machine learning algorithm in Ethereum. To prove the efficiency of BLOR, we simulated 100 oracles operating within a blockchain in 1000 observations and benchmarked it against several algorithms that vary by their degree of exploration and exploitation. It was found that BLOR prioritizes the newer oracles which hold less history and those with a fair balance of performance and price. Through a benchmarking experiment, BLOR proved a steady performance in selecting the successful oracles within a low to high noisy environment ranging, in average, from 80 to 90 percent out of the all selected oracles, whereas, other algorithms performance was ranged, in average, from 50 to 80 percent. BLOR had the most cost saving selection when the noise was lower. Overall, BLOR performed competitively better than the other algorithms, but at the cost of time. These research findings address our fourth research question which was about how to monitor the quality of the provided cloud services using the most qualified oracles.

8.2 Contributions

A summary of this thesis contributions are provided as follow:

1. We assessed the profitability of user ratings on cloud providers' income in a competitive on-line rating system. We addressed the problem of maximizing the providers' profit through a Stackelberg game model while adjusting the services' price and capacity based on the underlining users' demand.

2. We enabled providers to identify influential parameters on users demands and

captured the variations of users' demands in response to the changes of each parameter to enable scalability of cloud services and avoid under and over resources provisioning. Furthermore, our model helps in maintaining users' satisfaction and incentivizing them to provide good ratings for the providers.

3. We developed a two-stage game theoretical model to allow new and small cloud providers to compete against the existing and large ones and have a market share, which enables a productive cloud market industry that benefits the end-users.

4. We modeled and maximized users satisfaction using users' ratings by providing a continues service quality development. It is the first research that models a dynamic competition considering the quality of service among cloud providers. To ensure the continued validity of the optimality in the presence of changing internal or external factors, a post-optimality analysis is provided.

5. We elaborated a practical cooperative solution that any cloud provider can embrace regardless of their market position and trustworthiness through a fully distributed architecture with a democratic governance structure, called Cloudchain. To effectively enforce such a structure, Cloudchain proposes an innovative exploitation of blockchain to prompt and support interoperability and coopetition among the cloud providers over the public Ethereum network.

6. We incentivized the cloud providers and helped them make wise decisions about the utilization of Cloudchain by designing and solving a dynamic differential game. This game aims to maximize the profit of the Cloudchain members who cooperatively compete while their service demand is dynamically changing.

7. We developed a novel blockchain-based decentralized model that enjoys a

multi-agent structure, which allows us to introduce a quality verifier agent to ensure the cloud provider's compliance with the SLA. The interaction of an oracle within blockchain for monitoring purposes is innovative.

8. We formulated a three-player dynamic Stackelberg differential game in which players have to make choices about their control variables at various points in time. The optimal number of verification requests, capacity, and monitoring pricing are to be obtained through a Stackelberg differential game.

9. We put forward a new model using a Bayesian cost-dependent reputation model (BCRM) and knowledge gradient (KG) to find the most rewarding oracles. BCRM captures the behavior of the oracles elegantly, and KG unfolds the exploration/exploitation dilemma in multi-armed bandit with very low computational cost and high accuracy. We further proposed a framework to show how to employ the model within a blockchain where all the validators need to achieve a consensus. This framework incentivizes oracles to continuously act honestly and provide a fair balance of quality and price with minimal possibility of acting maliciously. To the best of our knowledge, there is no research that theoretically and practically implements a machine learning technique for blockchains and oracles.

The first and second contributions answered our first objective that sought to assess the profitability of user ratings on cloud providers' income and identify influential parameters on users demands in a competitive online rating system. The results are published in [79]. The third and fourth contributions are in response to our second objective that was to allow new and small cloud providers to compete against the existing and large ones and to maximize users satisfaction through a two-stage game theoretical model. The results are published in [81]. The fifth and sixth contributions

acknowledge our third objective that strive to advocate a fully distributed architecture using blockchain for cloud federation and to help the providers make wise decisions about the utilization of the blockchain-based federation. The results are published in [82]. The seventh, eighth and ninth contributions endeavored to achieve our forth objective that quested introduction of a new role of oracle through an innovative multi-agent framework and selecting the most qualified one/s to provide the service quality verification services using a Bandit-Bayesian Learning Oracle Reliability (BLOR) mechanism. The results are gathered within two research papers: one is published in [83], and the other one is under review of a refereed journal.

8.3 Directions for Future Work

The research contributions of this thesis filled out some of the important gaps in the current literature. However, considering the rapid advance of technology, there are still challenging problems to be explored. A summary of future research directions are provided as follow:

- Although game theory has been applied to real problems in different domains, the assumption that players are rational and have common knowledge so they aim at maximizing profit and minimizing cost is not always practical as shown by some experimental studies [12]. These experimental proposals demonstrated that in some cases, players consider in their decision-making other preferences than simply maximizing profits, for instance psychological, ideological, societal, or environmental preferences. Behavioral game theory could be utilized to investigate the behavior of cloud providers and customers' decision making using experimental data.

- This thesis theoretically analyzed the behavior of the small cloud providers

to gain a share in the market and verified that by simulating their behavior. Proving that our theoretical findings, supported by simulations involving real data, match experimental choices of real cloud providers in real settings is yet to be explored. To fully investigate the practical implication of our proposal, different cloud market players should be studied and analyzed if their behavior is as expected in theory, considering different considerations, for instance social, political, etc. This line of research is highly appealing as it has been demonstrated that real players play naturally towards the equilibrium solutions, in particular when the game is played many times so players gain experience and understand better the game [70].

- Supporting and deploying mobile-edge technologies are other interesting directions for further research toward the future of Cloud 2.0. In particular, two key issues can be focused on: security and computation offloading to tackle the problem of limited computational power, storage, and energy [36, 100].

- Application of blockchain and smart contracts in real-world scenarios is somehow neglected in the current literature. We investigated a specific application of public blockchain in cloud federation formation and many other applications could be explored by public or private blockchains. Private blockchains could be specifically challenging where we require to pay more attention to the roles and responsibilities of participants with different incentive mechanisms.

- There is a need to investigate different learning approaches to provide cloud players with better mechanisms to learn 1) the behavior of customers in order to increase their satisfaction and their ratings; and 2) better strategies to compete against different providers or collaborate within the Cloudchain platform.

Bibliography

[1] Ashraf Al Daoud, Sachin Agarwal, and Tansu Alpcan. Brief announcement: Cloud computing games: Pricing services of large data centers. In *DISC*, pages 309–310, 2009.

[2] Ehsan Khosrowshahi Asl, Jamal Bentahar, Hadi Otrok, and Rabeb Mizouni. Efficient community formation for web services. *IEEE Transactions on Services Computing*, 8(4):586–600, 2015.

[3] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *International Conference on Open and Big Data*, pages 25–30. IEEE, 2016.

[4] Anupam Kumar Bairagi, Md Golam Rabiul Alam, Ashis Talukder, Tran Hoang Nguyen, Choong Seon Hong, et al. An overlapping coalition formation approach to maximize payoffs in cloud computing environment. In *2016 International Conference on Information Networking (ICOIN)*, pages 324–329. IEEE, 2016.

[5] Kyrre Begnum, Mark Burgess, Tore M Jonassen, and Siri Fagernes. On the stability of adaptive service level agreements. *IEEE Transactions on Network and Service Management*, 3(1):13–21, 2006.

[6] Ryan Berryhill and Andreas Veneris. Astraea: A decentralized blockchain oracle. *IEEE Blockchain Technical Briefs*, 2019.

[7] Azer Bestavros and Orran Krieger. Toward an open cloud marketplace: Vision and first steps. *IEEE Internet Computing*, 18(1):72–77, 2014.

[8] Borja Bordel, Ramón Alcarria, and Tomás Robles. Denial of chain: Evaluation and prediction of a novel cyberattack in blockchain-supported systems. *Future Generation Computer Systems*, 116:426–439, 2021.

[9] Sarah Bouraga. A taxonomy of blockchain consensus protocols: A survey and classification framework. *Expert Systems with Applications*, 168:114384, 2021.

[10] Michèle Breton, Ramla Jarrar, and Georges Zaccour. A note on feedback sequential equilibria in a lanchester model with empirical application. *Management Science*, 52(5):804–811, 2006.

[11] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[12] Colin F. Camerer. *Behavioral game theory: Experiments in strategic interaction.* Princeton University Press, 2003.

[13] Victor Chang, Yen-Hung Kuo, and Muthu Ramachandran. Cloud computing adoption framework: A security framework for business clouds. *Future Generation Comp. Syst.*, 57:24–41, 2016.

[14] Haipeng Chen, Bo An, Dusit Niyato, Yeng Chai Soh, and Chuanyan Miao. Workload factoring and resource sharing via joint vertical and horizontal cloud federation networks. *IEEE Journal on Selected Areas in Communications*, 35(3):557–570, 2017.

[15] Junliang Chen, Chen Wang, Bing Bing Zhou, Lei Sun, Young Choon Lee, and Albert Y Zomaya. Tradeoffs between profit and customer satisfaction for service provisioning in the cloud. In *HPDC*, pages 229–238, 2011.

[16] Pei-Yu Chen, Yen-Chun Chou, and Robert J Kauffman. Community-based recommender systems: Analyzing business models from a systems operator's perspective. In *IEEE HICSS*, pages 1–10, 2009.

[17] Juan Pablo Romero Coronado and Jörn Altmann. Model for incentivizing cloud service federation. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 233–246. Springer, 2017.

[18] Valerio Di Valerio, Valeria Cardellini, and Francesco Lo Presti. Optimal pricing and service provisioning strategies in cloud systems: a stackelberg game approach. In *IEEE CLOUD*, pages 115–122, 2013.

[19] Z Diamadi, A Dubey, D Pleasance, and A Vora. Winning in the smb cloud: charting a path to success. *McKinsey, New York*, 2014.

[20] Wenjing Duan, Bin Gu, and Andrew B. Whinston. Do online reviews matter? an empirical investigation of panel data. *Decision Support Systems*, 45:1007 – 1016, 2008.

[21] Dave Durkee. Why cloud computing will never be free. *Commun. ACM*, 53(5):62–69, May 2010.

[22] Jehoshua Eliashberg and Abel P Jeuland. The impact of competitive entry in a developing market upon dynamic pricing strategies. *Marketing Science*, 5(1):20–36, 1986.

[23] Steve Ellis, Ari Juels, and Sergey Nazarov. Chainlink a decentralized oracle network. *White paper*, 11, March, 2017.

[24] Ming Fan, Subodha Kumar, and Andrew B Whinston. Short-term and long-term competition between providers of shrink-wrap software and software as a service. *European Journal of Operational Research*, 196(2):661–671, 2009.

[25] Yuan Feng, Baochun Li, and Bo Li. Price competition in an oligopoly market with multiple IaaS cloud providers. *IEEE Transactions on Computers*, 63(1):59–73, 2014.

[26] Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009.

[27] William George and Clément Lesaege. A smart contract oracle for approximating real-world, real number values. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[28] Naman Goel, Cyril van Schreven, Aris Filos-Ratsikas, and Boi Faltings. Infochain: A decentralized, trustless and transparent oracle on blockchain. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[29] Arthur S Goldberger. The interpretation and estimation of cobb-douglas functions. *Econometrica: Journal of the Econometric Society*, pages 464–472, 1968.

[30] Genaro J Gutierrez and Xiuli He. Life-cycle channel coordination issues in launching an innovative durable product. *Production and Operations Management*, 20(2):268–279, 2011.

[31] Makhlouf Hadji, Wajdi Louati, and Djamal Zeghlache. Constrained pricing for cloud resource allocation. In *IEEE NCA*, pages 359–365, 2011.

[32] Mohammad Mehedi Hassan, Abdulhameed Alelaiwi, and Atif Alamri. A dynamic and efficient coalition formation game in cloud federation for multimedia applications. In *Proceedings of the International Conference on Grid Computing and Applications (GCA)*, page 71. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[33] Xiuli He, Ashutosh Prasad, Suresh P Sethi, and Genaro J Gutierrez. A survey of stackelberg differential game models in supply and marketing channels. *Journal of Systems Science and Systems Engineering*, 16(4):385–413, 2007.

[34] Patrick Hoberg, Jan Wollersheim, and Helmut Krcmar. The business perspective on cloud computing - A literature review of research on cloud computing. In *18th Americas Conference on Information Systems, AMCIS 2012, Seattle, Washington, USA, August 9-11*, page Paper 5, 2012.

[35] Leslie M. Hocking. *Optimal control: An introduction to the theory with applications*. Oxford University Press, 1991.

[36] Binbin Huang, Zhongjin Li, Peng Tang, Shangguang Wang, Jun Zhao, Haiyang Hu, Wanqing Li, and Victor Chang. Security modeling and efficient computation offloading for service workflow in mobile edge computing. *Future Generation Computer Systems*, 2019.

[37] Richard Hull, Vishal S Batra, Yi-Min Chen, Alin Deutsch, Fenno F Terry Heath III, and Victor Vianu. Towards a shared ledger business collaboration language based on data-aware processes. In *International Conference on Service-Oriented Computing*, pages 18–36. Springer, 2016.

[38] Shweta Jain, Balakrishnan Narayanaswamy, and Y Narahari. A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[39] Yutao Jiao, Ping Wang, Dusit Niyato, and Zehui Xiong. Social welfare maximization auction in edge computing resource allocation for mobile blockchain. *arXiv preprint arXiv:1710.10595*, 2017.

[40] Toshiya Kaihara. *Supply Chain Management Based on Market Mechanism in Virtual Enterprise*, pages 399–408. Springer, 1999.

[41] Hamzeh Khazaei. *Performance modeling of cloud computing centers*. PhD thesis, University of Manitoba (Canada), 2013.

[42] Cinar Kilcioglu and Justin M Rao. Competition on price and quality in cloud computing. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1123–1132. International World Wide Web Conferences Steering Committee, 2016.

[43] Markus Klems, Jacob Eberhardt, Stefan Tai, Steffen Härtlein, Simon Buchholz, and Ahmed Tidjani. Trustless intermediation in blockchain-based decentralized service marketplaces. In *International Conference on Service-Oriented Computing*, pages 731–739. Springer, 2017.

[44] Petar Kochovski, Sandi Gec, Vlado Stankovski, Marko Bajec, and Pavel D. Drobintsev. Trust management in a blockchain based fog computing

platform with trustless smart oracles. *Future Generation Computer Systems*, 101:747–759, 2019.

[45] Daniel Kraft. Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications*, 9(2):397–413, 2016.

[46] Gabriella Laatikainen and Arto Ojala. Saas architecture and pricing models. In *IEEE SCC*, pages 597–604, 2014.

[47] Craig A Lee. Cloud federation management and beyond: Requirements, relevant standards, and gaps. *IEEE Cloud Computing*, 3(1):42–49, 2016.

[48] H. Li. Customer reviews in spectrum: Recommendation system in cognitive radio networks. In *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*, pages 1–9, 2010.

[49] Sin Kuang Lo, Xiwei Xu, Mark Staples, and Lina Yao. Reliability analysis for blockchain oracles. *Computers & Electrical Engineering*, 83:106582, 2020.

[50] Qinghua Lu, Xiwei Xu, Yue Liu, Ingo Weber, Liming Zhu, and Weishan Zhang. uBaaS: A unified blockchain as a service platform. *Future Generation Computer Systems*, 101:564–575, 2019.

[51] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269. ACM, 2016.

[52] L. Ma, K. Kaneko, S. Sharma, and K. Sakurai. Reliable decentralized oracle with mechanisms for verification and disputation. In *2019 Seventh International*

Symposium on Computing and Networking Workshops (CANDARW), pages 346–352, 2019.

[53] Jan Mendling, Ingo Weber, Wil Van Der Aalst, Jan Vom Brocke, Cristina Cabanillas, Florian Daniel, Søren Debois, Claudio Di Ciccio, Marlon Dumas, Schahram Dustdar, et al. Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 9(1):4, 2018.

[54] Carlos Mera-Gómez, Francisco Ramírez, Rami Bahsoon, and Rajkumar Buyya. A debt-aware learning approach for resource adaptations in cloud elasticity management. In *International Conference on Service-Oriented Computing*, pages 367–382. Springer, 2017.

[55] Andrew Miller. Permissioned and permissionless blockchains. In *Blockchain for Distributed Systems Security*, pages 193–204. John Wiley & Sons, Inc. Hoboken, NJ, USA, 2019.

[56] Subhas Chandra Misra and Arka Mondal. Identification of a company's suitability for the adoption of cloud computing and modelling its corresponding return on investment. *Mathematical and Computer Modelling*, 53(3):504 – 521, 2011. Telecommunications Software Engineering: Emerging Methods, Models and Tools.

[57] Vincenzo Morabito. Business innovation through blockchain. *Cham: Springer International Publishing*, 2017.

[58] Samar K Mukhopadhyay and Robert Setaputra. A dynamic model for optimal design quality and return policies. *European Journal of Operational Research*, 180(3):1144–1154, 2007.

[59] Carlos Müller, Hong-Linh Truong, Pablo Fernandez, Georgiana Copil, Antonio Ruiz-Cortés, and Schahram Dustdar. An elasticity-aware governance platform for cloud service delivery. In *IEEE SCC*, pages 74–81, 2016.

[60] Eric Münsing, Jonathan Mather, and Scott Moura. Blockchains for decentralized optimization of energy resources in microgrid networks. In *Conference on Control Technology and Applications (CCTA)*, pages 2164–2171. IEEE, 2017.

[61] Jayakrishnan Nair, Vijay G. Subramanian, and Adam Wierman. Provisioning of ad-supported cloud services: The role of competition. *Perform. Eval.*, 120:36–48, 2018.

[62] Guofang Nan, Zhifei Mao, Mei Yu, Minqiang Li, Honggang Wang, and Yan Zhang. Stackelberg game for bandwidth allocation in cloud-based wireless live-streaming social networks. *IEEE Systems Journal*, 8(1):256–267, 2014.

[63] Dusit Niyato, Athanasios V Vasilakos, and Zhu Kun. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 215–224. IEEE, 2011.

[64] Talal H Noor, Quan Z Sheng, Lina Yao, Schahram Dustdar, and Anne HH Ngu. Cloudarmor: Supporting reputation-based trust management for cloud services. *IEEE transactions on parallel and distributed systems*, 27(2):367–380, 2016.

[65] Ranjan Pal and Pan Hui. Economic models for cloud service markets: Pricing and capacity planning. *Theoretical Computer Science*, 496:113–124, 2013.

[66] Samir M Perlaza, Hamidou Tembine, Samson Lasaulce, and Mérouane Debbah. Quality-of-service provisioning in decentralized networks: A satisfaction

equilibrium approach. *IEEE Journal of Selected Topics in Signal Processing*, 6(2):104–116, 2012.

[67] Benay Ray, Avirup Saha, Sunirmal Khatua, and Sarbani Roy. Quality and profit assured trusted cloud federation formation: Game theory based approach. *IEEE Transactions on Services Computing*, 2018.

[68] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. challenges and opportunities. *Future Generation Computer Systems*, 88:173–190, 2018.

[69] Bruce Robinson and Chet Lakhani. Dynamic price models for new-product planning. *Management science*, 21(10):1113–1122, 1975.

[70] Larry Samuelson. Game theory in economics and beyond. *Journal of Economic Perspectives*, 30(4):107–130, 2016.

[71] Bo Shen, Yulong Shen, and Wen Ji. Profit optimization in service-oriented data market: A stackelberg game approach. *Future Generation Comp. Syst.*, 95:17–25, 2019.

[72] Roger Smith. Computing in the cloud. *Research-Technology Management*, 52(5):65–68, 2009.

[73] Amandeep Singh Sohal, Rajinder Sandhu, Sandeep K Sood, and Victor Chang. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. *Computers & Security*, 74:340–354, 2018.

[74] Basem Suleiman, Sherif Sakr, D. Ross Jeffery, and Anna Liu. On understanding the economics and elasticity challenges of deploying business applications on

public cloud infrastructure. *J. Internet Services and Applications*, 3(2):173–193, 2012.

[75] Gang Sun, Yayu Li, Yao Li, Dan Liao, and Victor Chang. Low-latency orchestration for workflow-oriented service function chain in edge computing. *Future Generation Computer Systems*, 85:116–128, 2018.

[76] Gang Sun, Dan Liao, Dongcheng Zhao, Zhili Sun, and Victor Chang. Towards provisioning hybrid virtual networks in federated cloud data centers. *Future Generation Computer Systems*, 87:457–469, 2018.

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

[78] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 01 1997.

[79] M. Taghavi, J. Bentahar, H. Otrok, O. A. Wahab, and A. Mourad. On the effects of user ratings on the profitability of cloud services. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 1–8, 2017.

[80] Mona Taghavi, Jamal Bentahar, Kaveh Bakhtiyari, and Chihab Hanachi. New insights towards developing recommender systems. *The Computer Journal*, pages 1–30, 2017.

[81] Mona Taghavi, Jamal Bentahar, and Hadi Otrok. Two-stage game theoretical framework for iaas market share dynamics. *Future Generation Computer Systems*, 102:173–189, 2020.

[82] Mona Taghavi, Jamal Bentahar, Hadi Otrok, and Kaveh Bakhtiyari. Cloudchain: A blockchain-based coopetition differential game model for cloud

computing. In *International Conference on Service-Oriented Computing*, pages 146–161. Springer, 2018.

[83] Mona Taghavi, Jamal Bentahar, Hadi Otrok, and Kaveh Bakhtiyari. A blockchain-based model for cloud service quality monitoring. *IEEE Transactions on Services Computing*, 13(2):276–288, 2020.

[84] Tram Truong-Huu and Chen-Khong Tham. A novel model for competition and cooperation among cloud providers. *IEEE Transactions on Cloud Computing*, 2(3):251–265, 2014.

[85] Sarah Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59(11):15–17, 2016.

[86] Sofía S Villar, Jack Bowden, and James Wason. Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 30(2):199, 2015.

[87] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. A stackelberg game for distributed formation of business-driven services communities. *Expert Syst. Appl.*, 45:359–372, 2016.

[88] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Trans. Services Computing*, 11(1):184–201, 2018.

[89] Shangguang Wang, Lei Sun, Qibo Sun, Jie Wei, and Fangchun Yang. Reputation measurement of cloud services based on unstable feedback ratings. *International Journal of Web and Grid Services*, 11(4):362–376, 2015.

[90] Leslie P. Willcocks, Will Venters, and Edgar A. Whitley. *Moving to the Cloud Corporation - How to face the challenges and harness the potential of cloud computing*. Palgrave Macmillan, 2014.

[91] Bolei Xu, Tao Qin, Guoping Qiu, and Tie-Yan Liu. Optimal pricing for the competitive and evolutionary cloud market. In *IJCAI*, pages 139–145, 2015.

[92] C. Xu, K. Wang, and M. Guo. Intelligent resource management in blockchain-based cloud datacenters. *IEEE Cloud Computing*, 4(6):50–59, 2017.

[93] Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran, and Shiping Chen. The blockchain as a software connector. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 182–191. IEEE, 2016.

[94] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *International Conference on Software Architecture (ICSA*, pages 243–252. IEEE, 2017.

[95] Bo Yang, Zhiyong Li, Shilong Jiang, and Keqin Li. Envy-free auction mechanism for vm pricing and allocation in clouds. *Future Generation Computer Systems*, 86:680 – 693, 2018.

[96] David Yermack. Corporate governance and blockchains. *Review of Finance*, 21(1):7–31, 2017.

[97] Qi Yu. Cloudrec: a framework for personalized service recommendation in the cloud. *Knowledge and Information Systems*, 43(2):417–443, 2015.

[98] Yugang Yu, George Q Huang, and Liang Liang. Stackelberg game-theoretic model for optimizing advertising, pricing and inventory policies in vendor managed inventory (vmi) production supply chains. *Computers & Industrial Engineering*, 57(1):368–382, 2009.

[99] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282, 2016.

[100] Feifei Zhang, Jidong Ge, Chifong Wong, Chuanyi Li, Xingguo Chen, Sheng Zhang, Bin Luo, He Zhang, and Victor Chang. Online learning offloading framework for heterogeneous mobile edge computing system. *Journal of Parallel and Distributed Computing*, 128:167–183, 2019.

[101] Jixian Zhang, Ning Xie, Xuejie Zhang, and Weidong Li. An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost. *Future Generation Computer Systems*, 89:286 – 299, 2018.

[102] Yu Zhang and Jiangtao Wen. The iot electric business model: Using blockchain technology for the internet of things. *Peer-to-Peer Networking and Applications*, 10(4):983–994, 2017.

[103] Laiping Zhao, Liangfu Lu, Zhou Jin, and Ce Yu. Online virtual machine placement for increasing cloud provider's revenue. *IEEE Transactions on Services Computing*, 10(2):273–285, 2017.

[104] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach D Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, 2019.