Dynamic Pricing Strategy for Maximizing Cloud Revenue

Fadi Alzhouri

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

August 2018

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Fadi Alzhouri**

Entitled: **Dynamic Pricing Strategy for Maximizing Cloud Revenue**

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

Doctor of Philosophy (Electrical and Computer Engineering)

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

Signed by the Final Examining Committee:

______________________ Chair
Dr. Zhibin Ye

______________________ External Examiner
Dr. Francois Gagnon

______________________ External to Program
Dr. Leila Kosseim

______________________ Examiner
Dr. Yan Liu

______________________ Examiner
Dr. Dongyu Qiu

______________________ Thesis Supervisor
Dr. Anjali Agarwal

Approved by ______________________
Dr. Mustafa Mehet Ali, Graduate Program Director

Thursday, August 20, 2018 ______________________
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Dynamic Pricing Strategy for Maximizing Cloud Revenue

Fadi Alzhouri, Ph.D.

Concordia University, 2018

The unexpected growth, flexibility and dynamism of information technology (IT) over the last decade has radically altered the civilization lifestyle and this boom continues as yet. Many nations have been competing to be forefront of this technological revolution, quite embracing the opportunities created by the advancements in this field in order to boost economy growth and to increase the accomplishments of everyday's life. Cloud computing is one of the most promising achievement of these advancements. However, it faces many challenges and barriers like any new industry. Managing and maximizing such a very complex system business revenue is of paramount importance. The wealth of the cloud protfolio comes from the proceeds of three main services: Infrastructure as a service (IaaS), Software as a service (SaaS), and Platform as a service (PaaS).

The Infrastructure as a Service (IaaS) cloud industry that relies on leasing virtual machines (VMs) has a significant portion of business values. Therefore many enterprises show frantic effort to capture the largest portion through the introducing of many different pricing models to satisfy not merely customers' demands but essentially providers' requirements. Indeed, one of the most challenging requirements is finding the dynamic equilibrium between two conflicting phenomena: underutilization and surging congestion. Spot instance has been presented as an elegant solution to overcome these situations aiming to gain more profits. However, previous studies on recent spot pricing schemes reveal an artificial pricing policy that does not comply with the dynamic nature of these phenomena.

In this thesis, we investigate dynamic pricing of stagnant resources so as to maximize cloud revenue. To achieve this task, we reveal the necessities and objectives that underlie the importance of adopting cloud providers to dynamic price model, analyze adopted dynamic pricing strategy for real cloud enterprises and create dynamic pricing model which could be a strategic pricing model for IaaS cloud providers to increase the marginal profit and also to overcome technical barriers simultaneously.

First, we formulate the maximum expected reward under discrete finite-horizon Markovian decisions and characterize model properties under optimum controlling conditions. The initial approach manages one class but multiple fares of virtual machines. For this purpose, the proposed approach leverages Markov decision processes, a number of properties under optimum controlling conditions that characterize a model's behaviour, and approximate stochastic dynamic programming using linear programming to create a practical model.

Second, our seminal work directs us to explore the most sensitive factors that drive price dynamism and to mitigate the high dimensionality of such a large-scale problem through conducting column generation. More specifically we employ a decomposition approach.

Third, we observe that most previous work tackled one class of virtual machines merely. Therefore, we extend our study to cover multiple classes of virtual machines. Intuitively, dynamic price of multiple classes model is much more efficient from one side but practically is more challenging from another side. Consequently, our approach of dynamic pricing can scale up or down the price efficiently and effectively according to stagnant resources and load threshold aims to maximize the IaaS cloud revenue.

Acknowledgments

I would like to express my gratitude to my supervisor, Professor Anjali Agarwal, for her guidance during my graduate studies. Thanks for giving me the chance to improve my academic knowledge. Further, thanks for teaching me precision and perfectionism skills in the work.

I would also like to express my appreciation towards my committee members Dr. Yan Liu, Dr. Dongyu Qiu and Dr. Leila Kosseim. I thank them for providing insightful comments and constant support throughout my study. The completion of this thesis would not have been possible without their efforts.

Dedication

I dedicate this thesis to

my love, my wife. I would like to thank her for standing beside me throughout my career and accomplishing this work. She has been my motivation for continuing to evolve my academic knowledge.

my loving parents and my brothers and sisters. Thank you for your complete support with my studies. My sincere thanks to my parents who have been my motivation to improve myself through all my walks of life.

my loving kids who sweeten my life with happiness and fun.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid evolution of technology led to the emergence of cloud computing as a delivery model, with a growing number of companies realizing ways to greatly improve efficiency. As the technology matures, there will be a frantic competition between organizations to deploy more elastic private, public and hybrid clouds, which promises to create an exciting level of investment in the field of information technology (IT). This will be due to a new way of delivering IT services and computing resources, which is, in turn, divided into the Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) layers.

Cloud computing environments has provided new opportunities to businesses that rely on the Information Technology (IT) industry allowing them to export their IT infrastructure's burdens and their expenses to cloud providers and avoid operational expenses, upgrades and maintenance costs of their own computing infrastructure. Using the Infrastructure as a Service (IaaS) cloud service model enables wide elasticity in terms of supply and demand for vendors and users, respectively. Users rent or buy as they use provisioned virtual machines (VMs) instead of physical machines (PMs). This is in contrast to the providers, who lease a variety of shared hardware, software, and IT services in the data center in the form of virtual entities to serve many users simultaneously as shown in Figure 1.0.1. IaaS cloud service providers provide important features to customers than on-premises data centres. These features allow the customer to focus on the development of their business rather than on the development of IT systems. The key features could be in short summarized as:

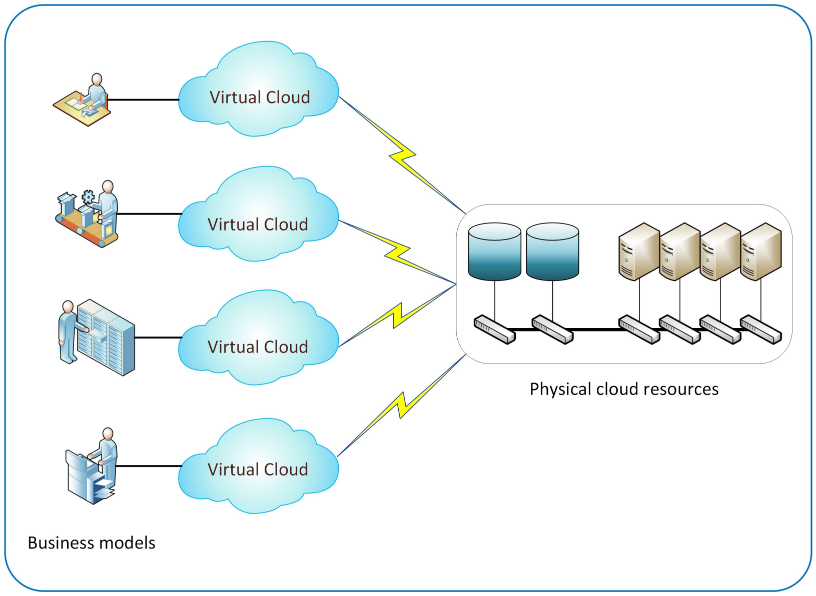


Figure 1.0.1: Cloud provider

- Pay for usage on demand and override hardware and maintenance expenses.
- Scale up and down the infrastructure as needed.
- Virtualize computing resources as well as organization administration.

However, the availability of IaaS cloud resources regardless of its features is of primary relevance in adopting the concept of using the IaaS cloud service. Consequently, cloud capacity, namely servers capacity, is a main challenge for cloud provider. So there must be a considerable capacity available to meet customer demands at any time, specifically at peak times. On the other hand, cloud providers news and studies state that these capacities are rarely used and the average utilization levels are under 20% [7]. Further, Uptime Institute research findings indicate that underutilized and stagnant servers account for 30% of local data centres' servers [14]. This issue, underutilization, inspires cloud providers to innovate new methods to drive utilization to higher levels. Multiple pricing schemes is one of these methods that has been reproduced from the economic area.

Many cloud computing enterprises, such as Amazon Elastic Compute Cloud (EC2) [2] and Microsoft Azure [6], adopt various pricing tiers that could be generally categorized into three types: on-demand, reserved, and spot price. These three tiers belong to two main pricing schemes: static and dynamic. The former is the dominant scheme in cloud computing, and it involves the provider charging the customer a fixed price per virtual machine per unit of time for long-term usage [16], [17], and [8]. The latter is a promising scheme, in which the provider charges the customer as long as the customer's bid equals or exceeds the instant price per time. Intuitively, these types have been inspired by the market economy applied in several fields, such as airlines, cargo, and renting services [80]. Pay-as-you-go and spot instances embodies the dynamic pricing strategy in the cloud computing economy. Not only the outstanding breakthrough that is achieved by dynamic pricing in revenue maximization but also the powerful advantage of mitigating congestion in the case of surge demand makes this area fertile ground for investigation in terms of the cloud computing. Indeed, IaaS cloud providers use spot market, namely dynamic price, to increase cloud utilization aims to maximize cloud revenue.

Primarily, technological enterprises present cloud computing as a cost-saving and agility technology that has a crucial role not only in the user's demand and budget but also in the revenue of the vendors. Intuitively, both of them aim to achieve maximum profit and minimum cost, taking advantage of research methods and models that arise in this area in order to find the optimal solution. The growth of cloud computing market and hence the raise competitors, the problem of maximizing cloud's profit becomes more critical. Thus, it will be remarkably important to evolve sophisticated mechanisms and schemes for pricing and managing cloud resources. While dynamic (or spot) pricing schemes strive to maximize IaaS cloud marginal revenue, dynamic resource management techniques target to minimize IaaS cloud resources costs using dynamic VM allocation.

It is worth noting that the best revenue management does not rely solely on a dynamic pricing scheme, because of the high risk incurred, but instead integrates elements of both static and dynamic schemes [33] and [44]. The role of dynamic pricing is to gain valuable revenue by enhancing the marginal profit through increasing the utilization of perishable resources, including CPUs, storage, and memory. The dynamic scenario automatically reduces the price to attract consumers when there is a plethora of stagnant/idle VMs and rare demand, and in the opposite case, it increases the price when VMs are rare and there is rising demand. In other words, the price obeys the law of supply and demand.

1.1 Problem Statement and Motivation

Cloud resources, such as CPU machine cycles, memory space, and network bandwidth, are perishable commodity and if they are not used during the offer period for sale or use at certain levels of utilization they waste associated cloud resource capacity and its operating expense as well. Cloud computing enterprises attempt to ensure that customers have sufficient resources from one side. However, on the other side, they aim to utilize incurred spare resources that are available most of the time [77] and [70] to achieve higher revenue. If a ready-to-sale VM is not consumed or requested during its offering time, the lost revenue cannot be claimed in future from IaaS cloud provider perspective. One subtle approach adopted by Amazon is Spot Instances [2]. In view of the novelty of this approach and its successes in other areas, namely airline booking, we dissect this problem and create a new spot model in addition to a new dynamic resource allocation tailored for spot market.

The problem can be superficially described as finding the optimum mapping from stochastic demands to stagnant virtual machines so as to hit the optimum spot price periodically and thus maximize the total expected revenue. It is clearly an intricate task from a theoretical and a practical point of view even before one delves into the sophisticated details. In this thesis, we adopt a dynamic pricing strategy using discrete-time stochastic Markov decision processes over a finite time horizon to address such a cloud revenue management problem. To this end, our work aims to overcome many significant challenges:

- The uncertainty of the arrival rate, also known as the demand to purchase, and the departure rate, also known as the demand to leave.
- The diversity of cloud states and the number of active and stagnant VMs.
- Making the right decision to accept or reject the request in real-time.
- Managing multiple fare types and many classes of VMs.
- Managing spare cloud resources for spot VMs instances.

1.2 Research Objectives

The ultimate objective of this research is to bring to light the role of spot pricing in managing cloud computing, produce new knowledge in this literature, and innovate a practical model that supports increasing cloud utilization and hence maximising cloud revenue. Consequently, this research will contribute to cloud providers adopting a dynamic pricing scheme to scale up or down the price depending on their spare resources and load threshold. Using this general framework, our goals focus on the following pertinent issues:

- Maximizing cloud revenue.
- Utilizing stagnant resources using multiple class of spot VMs.
- Designing a practical algorithm that converges with the expected optimal policy.
- Designing and implementing resources allocation approach for spot VMs.

1.3 Research Methodology

To achieve the aforementioned objectives, our methodological framework involves the following mechanisms:

1.3.1 Amazon Spot Instances Analysis

Indeed, the story started when Amazon launched the first dynamic pricing model in the cloud, Spot instances, to sell its spare computing capacity to customers who are willing to bid a price that at least meets the current spot price [3]. This stimulated many IaaS cloud providers to adopt a dynamic pricing strategy, and also encouraged researchers to investigate this trend in cloud. However, IaaS cloud providers do not declare much with regard to the adopted mechanism of dynamism until 2016. Furthermore, most discussions regarding Amazon's spot pricing demonstrate that it is artificially updated [22, 51, 52, 97]. Primarily, we explore Amazon Spot Instances and the studies about its pricing model in order to recognize closely the differences between spot prices for cloud providers and conclusions that have been presented in the state of the art of literature. The studies' findings refer to artificially controlled, Gaussian or Mixtures of Gaussian distribution, auction based, capacity-driven and not market-driven price. For further verification, we study Spot Instances pricing history for Amazon specifically for short intervals rather than long intervals to complement state of the art literature from on side and to accurately monitor the nature of price variation in spot market.

1.3.2 Dynamic Price Necessities

Although maximizing IaaS cloud revenue has been discussed before, specially its static price schemes, the literature still requires a survey that provides the readers with a comprehensive coverage of cloud's dynamic price. For instance, [22, 51, 68, 97] merely focused on detecting the rationale behind the dynamic technique that administers spot prices' changes for Amazon. However in this manuscript we highlight the objectives and necessities of dynamic price model for IaaS cloud providers and hence to create an active dynamic price model from engineering and economic perspectives alike. In order to reveal the underlying necessities behind utilizing the dynamic price as a pricing mechanism for VMs in IaaS cloud, first, we study the research related to cloud pricing schemes, more specifically dynamic pricing schemes, and its state of the art in this area. Then, We debate Amazon's motivations and those for other enterprises to adopt this scheme of pricing.

1.3.3 The Mechanism of Dynamism

In this phase after we answered the questions about what and why dynamic price attracts cloud providers to maximize their revenues, we have to investigate the key factors that play a vital role in pricing dynamism and hence formulate a price in terms of the most dominant and active parameters. Investigative research in this regard brought us to two main factors; stagnant resources and power consumption in addition to the elasticity. We deemed it wise to say that profits acquired by dynamic pricing scheme is a compensate for cloud elasticity, specifically after the new trend in cloud computing, namely Pay-as-you-go.

1.3.4 The Mechanism of Resource Allocation

The uncertainty of customers demands and their requirements impose cloud computing providers to adopt dynamic resource management in order to increase and decrease the capacity of supplied resources for sale periodically based on their needs [36, 41, 73]. IaaS resource management relies on estimating VMs workload. Indeed, cloud providers offers a variety of VM classes which differ by estimated workload or capacity. Therefore, developing effective cloud resource management mechanism is highly important to minimize resources costs. This task becomes more pressing in the dynamic pricing schemes, namely spot market. To this end, we leverage dynamic resource allocation based on the aggregate capacity to manage stagnant cloud resources. The dynamic resource allocation module plugs in with dynamic price scheme with a view to maximize cloud revenue.

1.4 Contributions

The main purpose of our work is to understand cloud computing spot price and thus create an efficient dynamic pricing strategy. To this end, this thesis target is to step forward through the following contributions:

- Provide a systematic study of the objectives underling dynamic price scheme in cloud computing environments.
- Understand the effects of both spot price and available capacity, the possible number of stagnant VMs, on demand. This contribution will be made through devising a pricing formula derived from economic domains that reflects the inverse proportionality between the price and the available capacity, or the remaining unutilized virtual machines. Obviously, the state of the system fluctuates between an abundance and a paucity of VMs, where the former necessitates a low price and the latter stimulates a high price.
- Formulate the maximum expected total revenue under discrete-time stochastic Markov decision processes over a finite horizon and employ probabilistic dynamic programming to solve such an optimization problem, considering the uncertainty of the arrival and departure rates

alike, in addition to respecting not merely one class but also multiple class of VMs, and underpin our formulation using optimum boundary conditions.

- Analyze and characterize the optimal solution for both one class and multiple classes.
- Aiming to make our approach practical, we approximate our stochastic dynamic programming approach using linear programming and develop an optimal pricing strategy algorithm based on linear programming relaxation in order to find the best decision policy in real-time, attempting to reach near-optimal expected revenue.
- Extend the previous point to reduce the time of matching the best decision which increases the efficiency of the cloud dynamic price system specifically in multiple classes scenario.
- Provide a dynamic resource allocation approach to manage spare cloud resources. The proposed approach strives for increasing each host utilization though VMs spot market.

1.5 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 reviews the related work and state of the art techniques in cloud dynamic price and dynamic resource management as well. In Chapter 3, we preface our approach by a detailed related background. We then present our proposed mathematical formulation and properties, linear approximation, and our CRM-DP algorithm for both one fare and multiple fares of one class in Chapter 4. In Chapter 5, we discuss the factors that drive price dynamism and mitigate the effect of high dimensionality on decision making time using column generation. We expand our approach to tackle multiple class of virtual machines (a fleet of VMs) in Chapter 6. In Chapter 7 we present a dynamic resource management approach for cloud spot market. This sort of resource allocation management works more effectively with the dynamic pricing system. To this end, we establish a model that combines dynamic resource allocation with dynamic price. Finally, Chapter 8 concludes this thesis, emphasizes its contributions and highlights some research that provides insights for future work.

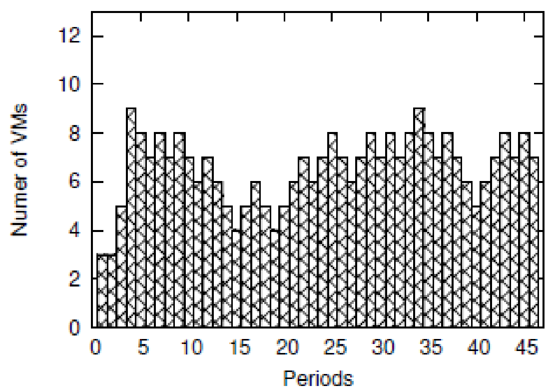
Chapter 2

Literature Review

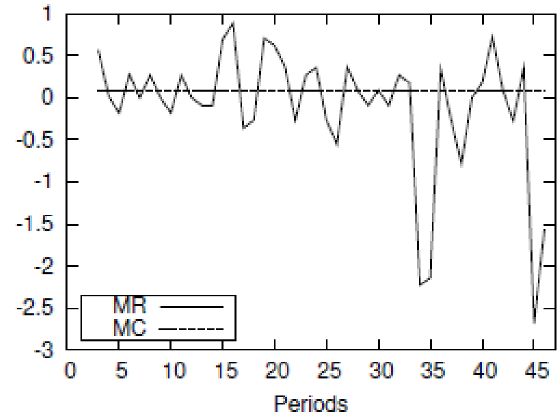
In this chapter we presents a survey of literature relevant to the area of dynamic price as well as dynamic resource allocation and provisioning to set the perspective for our contributions.

2.1 Cloud computing and spot price

The interesting concept, virtualization, in addition to rapid changes in hardware and software factors raise many significant challenges in cloud computing revenue management problem and vault it to be non-modest optimization problem. The pioneer cloud that takes on dynamic price as a solution for stagnant cloud resources was Amazon. The cloud spot market was launched in 2009 by Amazon's Spot Instances [20]. However, its adopted pricing strategy did not become known until late 2015. It is worth noting that cloud computing providers apply dynamic price only for one class of virtual machines via online but the deal about many classes should be arranged via contact. Amazon recently added dynamic price for multiple classes (it is called a fleet) [10]. This enticed researchers to analyze its spot pricing mechanism in detail and to broadly evaluate static and dynamic pricing schemes in cloud computing businesses aimed at maximizing cloud turnover values. While extensive research has examined static pricing schemes to increase profits through reducing the costs, little attention has focused on dynamic pricing schemes. For example, [51] analyzes the spot instances pricing for a year of four data centers of EC2 and statistically modeled spot price dynamics as a mixture of Gaussian distribution hourly and daily, which was a weak match for some instance types. Additionally, the accuracy of the results varies according to the number of components and the study is limited to a maximum of 4 components (VMs). [88] presents a microeconomic approach to meet the equilibrium point that satisfies, first, the consumer's budget and performance, and second, the vendor's profits. The cloud dynamically adjusts the number of VMs available for each user corresponding to the amount of money paid. However, the model absolutely assumes that cloud's response time and users' willingness to pay are inversely proportional and such a model is profitable when there is sufficient resources to satisfy all requests. Further, the proposed model is directed to reduce the cost bored by IaaS users through predicting the required numbers of VMs for users that satisfies their tasks as shown in Fig. 2.1.1 . By mimicking the natural selection process, [65] shows how a genetic algorithm model in a competitive cloud environment can evolve the price from naive to a convenient value to maximize profits. However, a brokers' platform is required and their model has some defects such as the convergence of revenue to a fixed value even if there are more tasks assigned by users, the experiment considered only 4 CPUs from cloud whereas the model discussed the revenue of all cloud resources and the time required to reach the most profitable price is not obviously identified. Likewise, some business mechanisms, such as negotiation, inspire [81] to adopt a PTN-price and time slot negotiations-mechanism. Its time slot utility function captures an agent's different scales of satisfaction for various time slots, using the "burst mode" algorithm. Whereas, auction induces [91] to model a dynamic auction, where customers may arrive to reserve a VM for many intervals, while a price dynamically changes according to proposed capacity allocations. Their proposed model wipes out the losses resulted from capacity allocation scheme, namely the auction only assign portion of IaaS resources for sale while the rest of resources wasting time without revenue and it also conflicts with the supply curve from economic perspective. [61] agrees and induces a heuristic approach to reduce the exponential complexity. Ultimately, this type of auctions is fruitful from a practical point of view merely when high arrival rates are met. [40] identifies the best hybrid cloud deployment and adopts a dynamic resource allocation model that accepts the request based on the utilization level in clouds. However, the model is restricted to identical physical machines clouds. [53] discusses the effect of remaining unused slots on maximizing the social welfare but not the revenue of cloud providers. In other words, it tries to find the balance between users themselves from one side and between users and providers from another side based



(a) VMs allocated per period



(b) Marginal Revenue & Cost

Figure 2.1.1: Evaluation in real private cloud reported in [88]

on hourly charged slots. Others try to implicate all pricing schemes, reserved, on-demand, and spot, in a single problem [85]. This approach has encountered many obstacles which resulted in rigorous assumptions and a reduction of the granularity. By dividing the capacity by 100, the time slot is one hour, one class of VM, and pseudo optimal algorithm. Needless to say, exploiting some characteristics of services would relatively increase returns, but the main issue remains the management of abstract VM as a commodity. From this perspective, [97] designs a revenue maximization system based on a continuous-time finite horizon with the stochastic demands of arrival and departure of an IaaS cloud. Then, [95] tackles the same problem as infinite horizon revenue maximization. There is actually a common or a joint perspective about all aforementioned papers which can be summarized by they missing the objective of dynamic price and for which quantity and type of products cloud be used. In this thesis, we maximize cloud revenue through dynamic discrimination of VMs' prices using discrete finite horizon Markovian decisions and hence, we utilize a column generation approach in order to increase decision-making speed. Our approach is approximately comparable with [97], except that it addresses multiple classes of VMs, derives some properties for both one class and multiple classes clouds that validate the functionality of our proposed model, and leverages a practical model using linear programming. These differences contribute to a more efficient performance from both the technical and practical perspectives.

2.2 Cloud computing and dynamic resource management

In this section, we list and discuss works related to resource allocation and management for IaaS cloud resources in terms of VMs. [71], [50], and [34] provide open source programs, namely Open Nebula, OpenStack, and CloudSim respectively, for the IaaS cloud to facilitate its understanding and study. This assists to study and analyze the techniques of resource provisioning in the cloud and then creates insights according to evaluations findings.

Low overall utilization levels, the surge in peak times and conditions of service are an important challenge for IaaS cloud providers. As resource management system, thus, is so substantial matter for cloud computing providers as it provides all the necessary reliability, efficiency, and elasticity. Therefore, cloud resource management systems extensively utilize virtualization technologies to increase resource utilization [36, 41, 73].

Yeo [98] presents resource-based admission control, server utilization metric, for grid computing. Moschakis [72] employs Gang scheduling to manage cloud resources in terms of VMs. The workload scheduling to the VMs is accomplished using two adaptive algorithms, first come first fit algorithm and a largest job first algorithm. Zhongyuan et al. [63] proposed priority-based resource scheduling algorithm called dynamic priority scheduling algorithm (DPSA) in cloud computing to solve service request scheduling problem. In DPSA, user requests are received, analyzed and categorized based on their particular requirements into task units to schedule directly on adequate resources and provide the effective service based on user request. Mandal and Khilar [67] proposed VM scheduler algorithm to reduce the time of allocation of VM to the server and to optimize the resource utilization. The algorithm represents the list of resources in a binary search tree (BST) instead of representing them in a queue. Thus, the resource utilization is improved and VM allocation time is reduced. In their algorithm, a BST is created for VM specifications and sent to VM scheduler. Servers are also listed in a BST. Using the BST of server and BST of VM, the VM scheduler will take the VM that has the maximum requirement and searches for a server which best fits the requirement of VM. Beloglazov and Buyya [30] proposed modified best fit decrease (MBFD) scheduling algorithm for VMs resource reallocation. In this algorithm, all VMs were sorted in decreasing order of their current CPU utilization. Each VM was allocated to a host that provided the least increase of the power consumption caused by the allocation. This heuristic algorithm which was based on the traditional greedy algorithm could optimize the allocation of VMs, but it was easy to fall into the local optimal and hard to achieve global optimal with a single point of the search strategy. Teng et al. [83] proposed equilibrium-based resources scheduling technique to predict the prospect price of resource without knowing competitors' bidding information. Nash equilibrium allocation proportion is received by users and fulfil deadline and budget constraints are met through the implementation on CloudSim. Tomás et al. [84] proposed a framework for VM placement that involves monitoring and profiling of applications to predict their behaviour and type of resource usage. The best location for the application to be deployed is determined via a smart overbooking scheduler. This approach is more suitable for software as a service (SaaS) cloud environment.

Zhen et al. [94] proposed virtualization based dynamic resource allocation mechanism to improve data center utilization. The predicted resources is based on the past behaviour of VMs using exponentially weighted moving average (EWMA) technique then a Skewness algorithm is used to estimate the disproportion in the multidimensional utilization of a processor through hotspot mitigation. This approach performs better in hotspot migration and load balancing, but live migration is not possible. [69] proposed a joint-VM provisioning approach in which multiple VMs are consolidated and provisioned based on an estimation of their aggregate resource need. They exploited statistical multiplexing among workload patterns of multiple VMs.

Chapter 3

Background

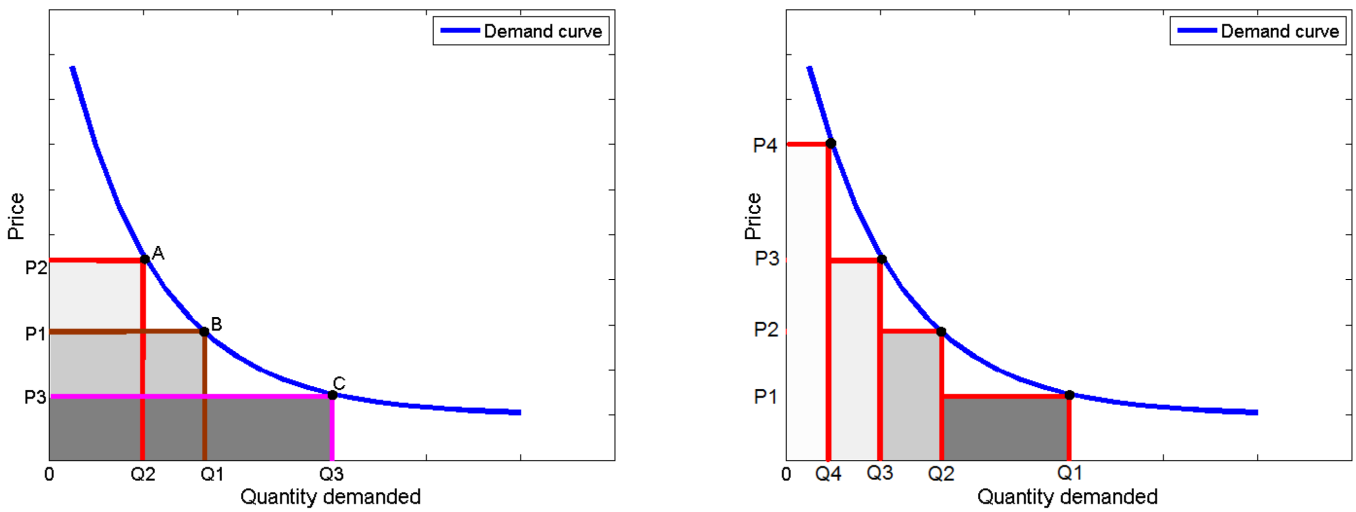
3.1 Overview

The principal objective in this chapter is to construct the necessary and essential knowledge required to design a dynamic pricing scheme, realize cloud spot market, and develop a dynamic resource allocation mechanism for spot market from one side and aims at simplifying the highlighted concepts in this manuscript. For this reason, we present the following three issues:

- Price vs Demand.
- Markov Decision Processes.
- Column Generation.

3.2 Price vs Demand

Price and demand are among of the most fundamental concepts of economics and the basic criteria for the market law. The demand, or quantity demanded, for a product is the amount that customers are willing to buy at a certain price and time apart from other factors. The law of demand states that the higher the price of a product, the lower the quantity demanded by customers, and vice versa. Indeed, this is true if all other affecting factors remain equal. The demand curve related to certain product could be created from the market history by using horizontal summation process,



(a) Demand Law

(b) Total Revenue

Figure 3.2.1: Price demand

namely adding the quantity demanded by all customers at each price. The demand curve, also known as the willingness-to-pay curve, shown in Fig. 3.2.1a, depicts the relationship between price and quantity demanded. Intuitively, $the\ total\ expected\ revenue = price \times quantity$, which represents the area below the curve. Let us assume that the price for each VM is P1. Then, Customer B is willing to buy Q1 VMs, the quantity demanded, and the revenue acquired by pricing a VM at P1 will be $R1 = P1 \times Q1$, which is represented by the brown frontier area. However, if the cloud increases the VM price to P2, then it will sell Q2 VMs to Customer A resulting in $R2 = P2 \times Q2$, which is represented in the red frontier area. Thus, if it decreases the price to P3, Q3 VMs will be consumed by Customer C. Given this, we clearly can show that a changing price, or multi-fare, scenario is a good approach to capture most of the expected total revenue that is depicted in Fig. 3.2.1b in the form of four cascaded bars. The total expected revenue could be even higher depending on the price segmentation, which is beyond the scope of our research. The interesting point that should be mentioned is the effects of other factors rather than price and quantity on demand curve which can be briefly drawn by shifting the entire demand curve to the right or to the left in case of increase or decrease in demand respectively.

3.2.1 Pricing Mechanisms in the Cloud

In this section we present a set of pricing mechanisms in the cloud. These mechanisms vary according to the criteria of the cloud service provider and the type of service itself as shown in Fig. 3.2.2 [76].

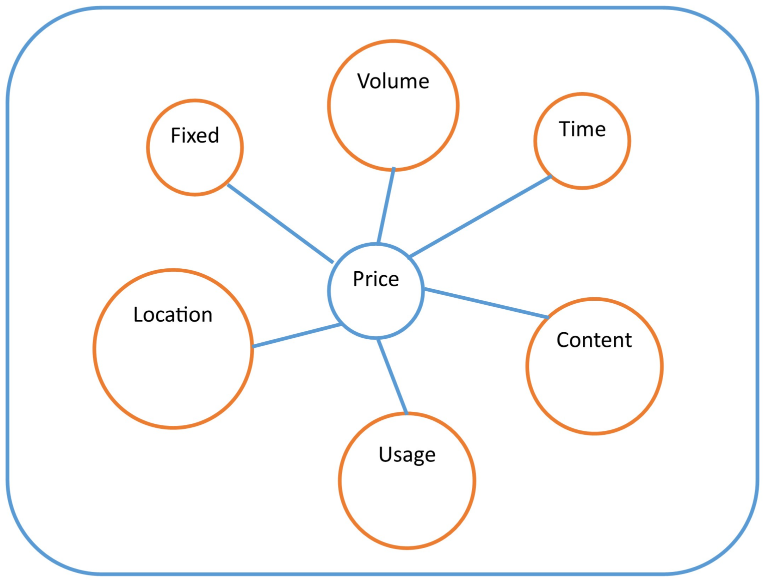


Figure 3.2.2: Pricing mechanisms

- Time based price: the price is based on time duration of use.
- Volume based price: the price is based on the value volume of a metric (e.g. data volume per application).
- Fixed : the price is fixed for a specified time unit.
- Usage based price: the price is based on the usage of the service for a time slots (e.g. pay-as-you-go).
- Content based price: the price is based on the type of the content of the service.
- Location based price: the price is based on the location-space of the service.

3.2.2 Pricing Schemes in the Cloud

Cloud computing providers adopt many pricing schemes to sale their computing commodity. These schemes can be categorized into three basic plans: dynamic plan, reserved plan, and demand plan, from a IaaS cloud providers' perspective (i.e. as shown in Fig. 3.2.3).

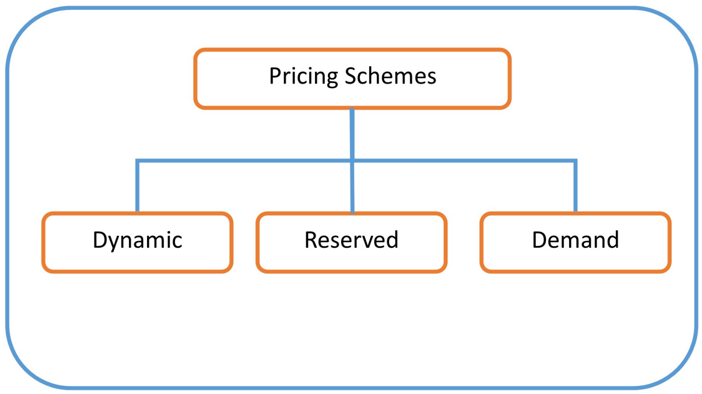


Figure 3.2.3: Pricing schemes

- Dynamic plan: Recently, IaaS cloud providers have provided computing services on a pay-as-yo-go basis. In this plan, the provider quantifies the time units of the services that the customer use, and charge customers depending on it. This pricing plan of billing of service consumption is like the utility services such as mobile usage. However, cloud providers often charge for services on a dynamic-rate basis not fixed-rate basis as in another areas. This thesis aims to study and analyze this pricing plan particularly, then attempts to introduce a new dynamic pricing scheme aiming at increasing IaaS cloud revenues.

- Reserved plan: This pricing plan allows customers to pay based on a subscription fee to access cloud service for a predefined time period, mostly long time periods. The service is also on a pay-as-yo-go basis but with upfront commitment and fixed-rate basis. The idea behind reserved pricing plan is that cloud providers insure the availability, confidence, and quality of the service, namely SLA, for their customers.

- Demand plan: In the demand plan, the price for a service is determined based on the level of demand. In the beginning of the session, the level of demand is evaluated and then the price is determined. This plan is similar to the dynamic pricing plan but is non-interruptible.

3.3 Cloud Revenue Management

The large financial boom that has been achieved by cloud computing is expected to continue growing at a powerful rate over the course of the next years. Therefore, investing in promising technology-delivering service companies is believed to be the next tech bubble. Actually, many enterprises claim that they gain money from the robust growth of this industry. For example, Amazon Web Services [7] earns about $6 billion per year, and so does Microsoft at $6.3 billion in 2014 [5]. Gartner, a well-known firm in technology industry research, anticipates that the highest growth in cloud computing market will be for IaaS services up to $71.5 billion in 2020 [9]. Furthermore, Synergy reports that investments in IaaS cloud service earned $20 billion in revenues for the year 2015 and witnessed a market growth of 28% [1]. Despite these economic incentives, vast critical issues exist in this fresh technology industry from cloud providers' perspective. Maximizing cloud revenue is one of the most critical challenges faced by cloud providers, especially when taking utilization levels into account. We do not claim to expose this shortcoming, indeed the most challenging factor for the IaaS cloud industry is finding the dynamic equilibrium between two of conflicting phenomena: underutilization and surging congestion. Amazon disclosed that utilization levels in 2015 were often under 20% [7]. In general, the massive efforts toward addressing this problem can be categorized as i) minimizing operational costs (e.g. power consumption and cooling systems) [35,42,82], and ii) maximizing cloud revenue by creating more protable pricing models [2,61,87,90]. While price has a significant impact on customers' behaviour as it directly affects their budgets, it remains a major concern for cloud providers, as it drives the growth of their profits by allowing them to maximize IaaS cloud revenue. IaaS cloud providers are stimulated to adopt a dynamic pricing strategy for managing perishable spare VMs and also to increase cloud utilization.

3.4 Markov Decision Processes

Thus far, we have focused on pricing vs demand and cloud revenue from an economical point of view but building a strong foundation calls for mathematical notions and notations. In particular, this subsection presents a probability model for processes that evolve through a finite time horizon using a stochastic method. This model is called a Markov chain and it is memoryless. The process as it evolves in the future is independent of the process as it was in the past and dependent merely on the current state. Thus, a Markov chain is a stochastic process $\{X_t\}$ that satisfies the following property:

$$\begin{aligned} P(X_{t+1} = j | X_t = i, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0) &= P(X_{t+1} = j | X_t = i) \\ &= P_{ij} \\ &\quad \textit{for all } t \geq 0 \textit{ and } j, i, i_{t-1}, \ldots, i_0 \in S, \end{aligned}$$

where S is the set of all possible states and P_{ij} is a transition probability from state i to state j in any step $t \in \{1, 2, \ldots, T\}$ and the matrix that represents one-step transition probability for all states is denoted as $\mathbf{P} = (P_{ij})_{i,j \in S}$ and is constrained by:

$$\sum_{j \in S} P_{ij} = 1, \qquad \forall i \in S.$$

Given a sequence of actions that control state transitions and dynamic programming concepts, this process is called a finite-horizon Markov decision process, which is a robust tool developed by Bellman [28] to model numerous real-world problems. In our research framework, we can, in general, describe a stochastic Markov decision model using the following basic form of Bellman's equation:

$$f_t(X_t) = \max_{a_t \in A} (R_t(X_t, a_t) + \mathbb{E}\, f_{t+1}(X_{t+1} = H(X_t, a_t, D_{t+1})))$$

which states that the objective function f_t, or a value function, of an initial state X_t at stage t is evaluated depending on decision a_t that maximizes the combination of consequent reward R_t and the expected value f_{t+1} of landing in state X_{t+1}, where a transition function H determines state X_{t+1} depending on the current state X_t, a decision a_t, and a probabilistic exogenous incident D_{t+1}, which is, in our case, the demand. This expectation can be represented using a transition matrix associated with each decision. It is worth noting that, the nature of our research calls for using a Poisson probability distribution to generate such a matrix. Moreover, a backward recursion could be used to solve this problem instead of traversing all possible sequences of decisions.

3.5 Column Generation

Column generation is subtle approach in computational optimization to settle a mathematical program by iteratively boosting the most promising variables of the model. More specifically, column generation is an algorithm for solving large-scale linear programming problems [32]. The column generation umbrella covers a variety of methods with some variations and hence we operate Dating-Wolfe decomposition in the proposed dynamic price algorithm, we introduce the following brief description about its rationale. The constraints of a linear programming (LP) problem is partitioned into two sets, the first is called Master constraints and the second is called subproblem constraints. This means that the original large problem can be decomposed into a master problem with a large number of variables and subproblems. The later generates new extreme point variables (new column) for master problem which provides prices to the subproblems and, hence the name column generation. The subproblem (is called pricing problem) evaluates if there is a new column with a positive reduced cost to add it to the master problem, and so forth. Fig. 3.5.1 depicts the structure of the general process. It worth mentioning that the cutting stock problem is a well known

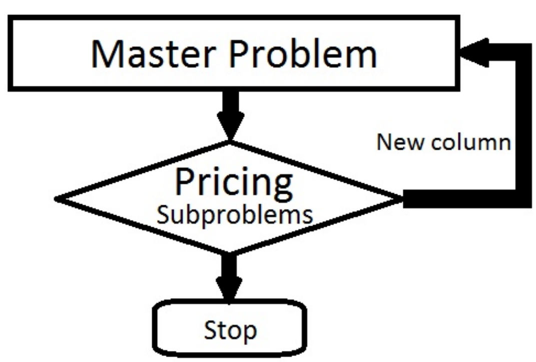


Figure 3.5.1: A column generation algorithm.

example regarding column generation approach. Basically, the cutting stock problem is an integer programming (IP) problem. However, this type of problems is known to be of NP-hard type, and hence, the problem is formulated as an LP by relaxing the integer constraint. Ultimately, the results of LP are rounded to integer values. The cutting patterns that can be generated from the stock lengths represent the basis matrix . When the number of columns, namely cutting patterns, is very large, this makes the task of enumerating all feasible patterns impractical. Therefore a set of feasible patterns represents the basis for the simplex method to resolve for the dual variables.

3.5.1 Cutting Stock Problem

The problem consists of cutting big available stock lengths into small patterns to meet customer orders while minimizing the number of stock materials used and consequently the cost. A given instance assigns the length of the stock material, the length of the desired cuts, and the demand for each cut type. Clearly these variables are integers variables and the notation for them is defined as following:

W: the length of stock material

w_i: the length of cut i, $i = (1, \ldots, n)$

b_i: the quantity demanded of each cut

x_j: number of times pattern j is used, $j = (1, \ldots, m)$

C: the cost of stock material

The model is formulated as:

$$min \sum_{j=1}^{m} C x_j \tag{1}$$

$$S.t. \quad \sum_{j=1}^{m} a_{ij} x_j \geqslant b_i, \quad \forall i \tag{2}$$

$$x_j \geqslant 0 \quad \forall j. \tag{3}$$

Assume that $C = 1$ for the sake simplicity. Consequently, The objective function (1), the total cost of stock lengths, becomes $min \sum_{j=1}^{m} x_j$. Constraint (2) implies that each quantity demanded b_i must be fulfilled. Furthermore, assuming x_j is an integer variable makes this problem of type NP-hard. Hence relaxing x_j makes this problem belongs to linear programming problems which can be easily solved by using simplex method. The solution results from this problem can be rounded to output a feasible solution for the original problem. The solution for this relaxed problem will not output an optimal solution but it will give approximate optimal solution. Thus adding a few stock materials by rounding up the relaxed solution will not have a huge impact on the desired solution. In literature the previous problem is called master problem. The dual of this problem is given as:

$$Max \sum_{i=1}^{n} a_i y_i \tag{4}$$

$$S.t. \quad \sum_{i=1}^{n} a_i w_i \leqslant W, \tag{5}$$

$$a_i \geqslant 0 \quad and \quad integer \quad \forall i. \tag{6}$$

The solution for the above problem is derived as the set $(a_1, a_2, \ldots, a_n)$ that maximizes (4). If such a pattern (or column, hence the name) cannot be generated then the iteration terminates and current solution is optimal. Otherwise, this pattern enters basis.

Example: $W = 20$: the length of stock material.

$C = 1$: the cost of stock material.

$n = 4$: the number of cuts.

The cut of length $w_1 = 9$ is demanded $b_1 = 511$.

The cut of length $w_2 = 8$ is demanded $b_2 = 301$.

The cut of length $w_3 = 7$ is demanded $b_3 = 263$.

The cut of length $w_4 = 6$ is demanded $b_4 = 383$.

The optimal solution is depicted in Table 3.1. It is clear that the total stock materials used is

Cut	P1=255.5	P2=87.625	P3=131.5	P4=125.75
9	2	0	0	0
8	0	2	0	1
7	0	0	2	0
6	0	0	1	2

Table 3.1: Non-integer optimal results

600.375. However, rounding up the above results gives the integer solution $(P1 = 256, P2 =$

$88, P3 = 132, P4 = 126)$. Consequently, the total stock materials used is 602, which is the optimal solution, instead of the optimal 600.375.

3.6 Summary

We provided some principles derived from economic area which present basic knowledge of the relationship between the price and quantity of commodity that customer is willing to purchase. These principles are the first step in our investigation to understand how pricing schemes work and to build upon them. Then we listed the major pricing mechanisms and schemes used in the cloud. Research, studies and news of cloud revenue management indicate promising profits through IaaS cloud investment, but it needs to develop current pricing schemes and mechanisms to increase these returns. Our approach falls mainly in the maximizing IaaS cloud revenue research area. Obviously, uncovering the obstacles coping IaaS cloud providers, and the techniques behind it can help in innovating new methods and hence can solve the problem. Therefore, we presented a summary of Markov Decision Processes and Column Generation techniques, which we adopted in this thesis to develop dynamic price scheme specifically targeting spot VMs.

Chapter 4

One class dynamic price scheme

4.1 One Class System Model

In this chapter, we elaborate on the model, implementation and validation of dynamic pricing scheme for maximizing IaaS cloud revenue considering merely one class of VMs on-the-shelf.

4.1.1 Modelling Assumptions

The model represents Infrastructure as a Service (IaaS) cloud computing in which the provider rents physical machines (PMs) of a data center in terms of virtual machines (VMs). Indeed, cloud providers do not reveal much information about the actual size of their data centers, but according to [26] and [18], the estimated size is more than 10,000 servers. We believe the best revenue management is achieved by the mutual cooperation between the static and dynamic schemes, whereas on-demand pricing provides solid and stable revenue, while spot pricing strives to increase the expected profit. As shown in Fig. 4.1.1, the state of the system can be described, at any time, as a set of active VMs, or rented, which rely on a set of on-PMs in addition to a set of off-PMs. These active VMs consume considerable capacity of ”on” physical machines, but there is still a notable amount that can be considered as stagnant VMs. In this perspective, we use the term ”stagnant VMs” to refer to those unrented VMs that still consume operational costs. It is worth noting that when there are high arrival demands, an instance price greater than the maximum static price, and no extra VMs in the idle pool in the previous horizon, the system can turn on new PMs in the next horizon if they are available. On the contrary, conducting the [54], [74], and [78] approach in our model, dynamic pricing can be used as a key controlling tool to attenuate cloud congestion. In such a case, increasing the instance price simultaneously increases the departure rate. In particular, the objective of this thesis is to find the best utilization of these stagnant VMs to maximize profit of the cloud within a given horizon. Table 4.1 presents a description of the main notations and variables used in our problem formulation. For example, assume a cloud provider has $|N|$ number of VMs

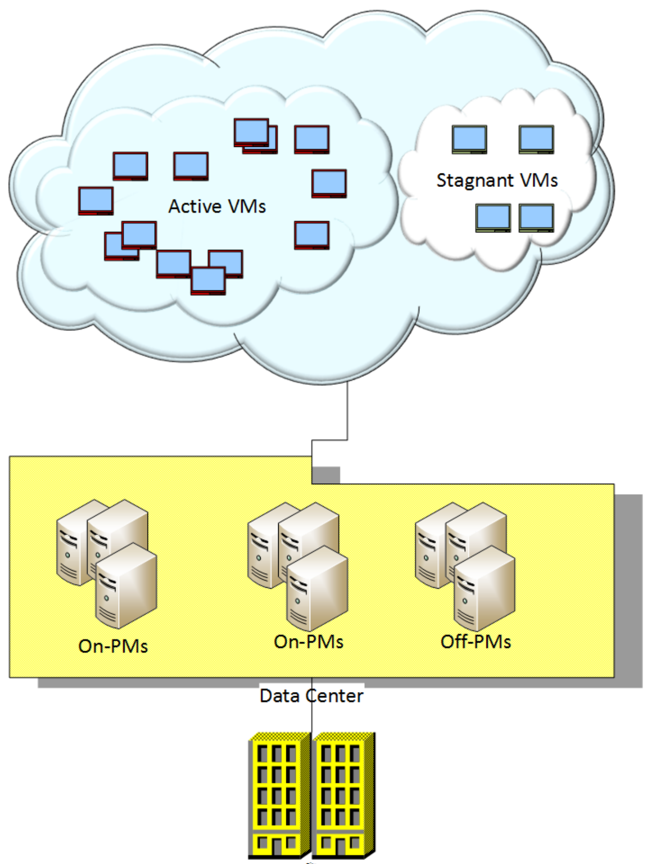


Figure 4.1.1: A system model.

available in a certain data center. $c_j(t) = \lfloor |N|/j \rfloor$ represents the available number of stagnant VMs of fare j at time t, $c_j(t) \in C \triangleq \{0, 1, \ldots, |N|\}$. Based on these notations, the state of the system, or the pool of stagnant VMs, at any given interval t is denoted as $\mathbf{C}(t) = \{c_1(t), c_2(t), ..., c_{|N|}(t)\}$. Assume T is the length of horizon H, then H can be calculated as shown in equation 7.

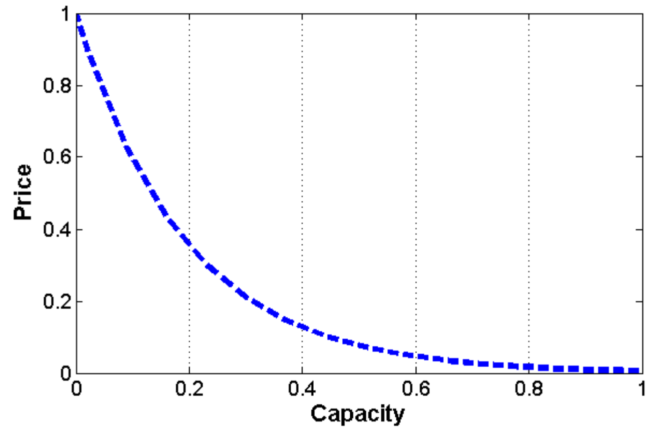
Table 4.1: Main notations and variables

Notation	Description
$\|N\|$	number of VMs available for sale.
$c_j(t)$	number of stagnant VMs of fare j at time t.
T	time length of sale horizon H.
$\triangle t$	shortest time interval in which only one incident occur.
R_j	fare of type j.
$\rho_j(t)$	price of type j at time t.
$\lambda_j(\rho, t)$	arrival rate for type j of price ρ at time t.
$\mu_j(\rho, t)$	departure rate for type j of price ρ at time t.
$\mathbf{A}^j$	request of j type of VMs.
u_j	decision to accept or reject a request for fare j.
$f(C, t)$	total expected revenue of available resources C at time t.
γ	mean arrival rate in the horizon.
β	mean departure rate in the horizon.

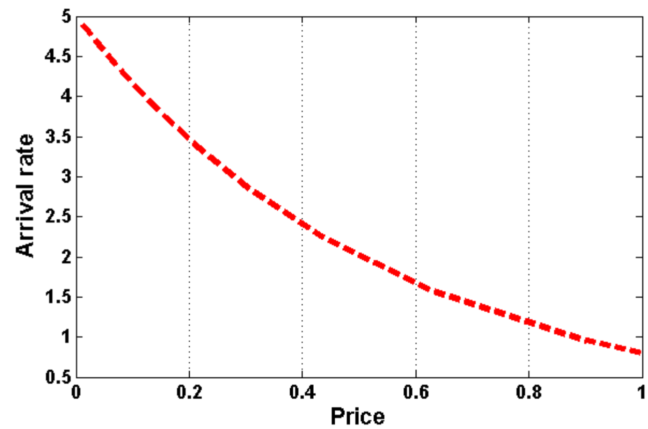
$$H \triangleq \sum_{s=1}^{T} \triangle t_s \tag{7}$$

$\triangle t_s$ is the time slot, and this epoch must be small enough to allow only for one incident, or one request, to occur with obedience to the Poisson process condition. Practically, the demand includes fare type $j \in \{1, 2, ..., |N|\}$ that represents a class of services and directly contributes R_j fare to cloud revenue, then the entire set of fare types $\mathbf{R} = \{R_1, R_2, ..., R_{|N|}\}$, and without a loss of generality let order fare types $R_1 < R_2 < ... < R_{|N|}$.

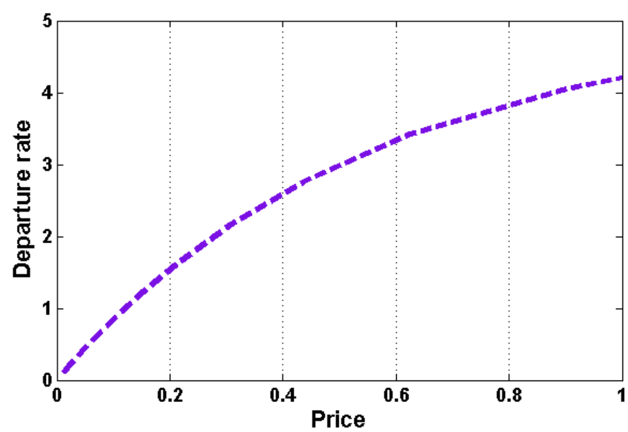
Let $\rho_j(t) \in [0, \rho_{max}]$ denote the price of type j at time t, where $\rho_{max} \leq 1$. We define price as an exponent function of capacity and time. For example, $\rho_j(t) = e^{-c_j(t)\epsilon}$, where ϵ is a calibration parameter related to time, and such a definition seems convenient in our case. The empty set of j stagnant type, $c_j(t) = 0$, allows the price to grow up to the maximum value as an implicit condition that imposes the cloud to price at $\rho_j(t) = \rho_{max} = 1$ when stock is zero. In contrast, the cloud forces the price to be zero, $\rho_j(t) = 0$, in case of an abundance of capacity $c_j(t) = 1$. Fig. 4.1.2a shows the impact of capacity over the course of pricing operation that varies within the interval [0,1]. It worth noting that the values of capacity and price are normalized and cover the complete range from the null to the maximum value.



(a) Price versus capacity



(b) Arrival rate versus price



(c) Departure rate versus price

Figure 4.1.2: Price versus capacity and arrival and departure rate versus price

We can assume, without loss of generality, that demand is a function of arrival or departure rates and time and that the accepting and rejecting processes do not affect these rates. The arrival rate related to type j of price ρ at time t is $\lambda_j(\rho, t)$

$$\lambda_j(\rho, t) = \gamma e^{-\alpha \rho_j(t)} \tag{8}$$

and the departure rate is $\mu_j(\rho, t)$

$$\mu_j(\rho, t) = \beta(1 - e^{-\alpha \rho_j(t)}), \tag{9}$$

where the mean arrival rate and the mean departure rate in the current horizon $\gamma, \beta \geq 0$ and scaling parameter $\alpha \geq 1$. Note that when $\rho_j(t) = 0$, the lowest price, then $\lambda_j(0, t) = \gamma$ and $\mu_j(0, t) = 0$, and theoretically there is no incentive to leave the system. But when $\rho_j(t) = 1$, the maximum price, then $\lambda_j(1, t) \simeq 0$, indicting there is no motivation to utilize any VM, and $\mu_j(1, t) \simeq \beta$ forcing customers to leave cloud services. Clearly $\alpha = 1$ in (8) and (9). An example of such functions shown in Fig. 4.1.2b and 4.1.2c depict the harmony with the law of demand curve defined in microeconomics principles [64] and [47].

4.1.2 Model Formulation

The problem of dynamic pricing strategy can be formulated as a finite horizon Markov decision process under uncertainty. We consider a discrete time horizon in which at any given interval and any possible state of the Markov chain, the system chooses the optimal decision, to sell or to wait, by traversing all stochastic possibilities of subsequent rewards up to the end of horizon, with a view toward gaining the maximum expected revenues.

At the beginning of each horizon $t = 1$, non-investing capacities of resources represent a finite space $\mathbf{C}$ supplied by cloud for sale $(\mathbf{C}, t)$ and a type request $\mathbf{A}$, demanded by customer. If the system accepts the request of type j, it moves to the state $(\mathbf{C} - \mathbf{A}^j, t+1)$ in the next epoch. Whereas, in case of rejecting the request, the state becomes $(\mathbf{C}, t+1)$. On the other hand, the system moves to the state $(\mathbf{C}+\mathbf{A}^j, t+1)$ as a result of releasing type j VM from the customer. At each time interval, Poisson incurs only one request of arrival and departure, if any, with that being a request to hold a type j, which arrives in time t with probability $p_j(t)$, or to release it with probability $q_j(t)$. Instantly, a binary decision u_j, to sell or to wait, must be taken according to the current state and offer R_j of type j customer. The decisions chain up to the end of horizon $t = T$ aims to maximize the total expected revenue $f(C,t)$ derived from Hamilton-Jacobi-Bellman's equations as follows:

$$\begin{aligned} f(C,t) \;&=\; \max_{\vec{u}\in U}\{\sum_{j=1}^{n} p_j(t)(R_j u_j + f(C - A^j u_j, t+1)) \\ &+\; \sum_{j=1}^{n} q_j(t) f(C + A^j, t+1) \\ &+\; (1 - (\sum_{j=1}^{n} p_j(t) + \sum_{j=1}^{n} q_j(t))) f(C, t+1)\}, \quad \forall t \quad C \in \mathbf{C}(t) \end{aligned} \tag{10}$$

subject to the terminal conditions:

$$\begin{aligned} &f(C,0) \triangleq 0, \\ &f(C,T+1) \triangleq 0, \; and \\ &f(0,t) \triangleq 0, \qquad \forall t, \, \forall C \end{aligned}$$

Before the beginning of horizon, $t = 0$, and after the end of it, $t = T + 1$, the revenue must be updated to zero as well when all resources are exhausted, $C = 0$, through any time.

Clearly, the maximization process implies a decision to satisfy

$$u_j = \begin{cases} 1 & if \; R_j + f(C - A^j, t+1) \geq f(C, t+1) \\ 0 & otherwise, \end{cases} \tag{11}$$

to judge the more promising action, either to sell, incrementing the profit by R_j or to wait. The value of the objective function (10) from a given state can be briefly formulated to

$$\begin{aligned} f(C,t) &= f(C,t+1) + \sum_{j=1}^{n} q_j(t) \triangle f(C + A^j, t+1) \\ &+ \max_{\vec{u} \in U} \{\sum_{j=1}^{n} p_j(t)(R_j u_j - \triangle f(C,t+1)), 0\}, \quad \forall t \quad C \in \mathbf{C}(t). \end{aligned} \tag{12}$$

Whereas

$$\triangle f(C,t+1) = f(C,t+1) - f(C - A^j, t+1) \tag{13}$$

is the opportunity cost of available capacity $\mathbf{C}$ incurred by incident decision (11). It is noteworthy that the optimal policy is to sell whenever a bid R_j overrides its associated opportunity cost (13). The implicit effect of releasing resources on the grasped revenue is marked by

$$\triangle f(C + A^j, t+1) = f(C + A^j, t+1) - f(C, t+1) \tag{14}$$

To achieve the cloud's objective (12), which is to maximize the total expected rewards, the system finds

$$u_j^* \in \operatorname*{argmax}_{\vec{u} \in U} \{\sum_{j=1}^{n} p_j(t)(R_j u_j - \triangle f(C,t+1))\} \tag{15}$$

every interval t resulting in an optimal n-stage control policy $\pi^* = \{u_1^*, u_2^*, ..., u_n^*\}$ that yields the maximum expected revenue

$$f^*(C,t) \triangleq \sup_{\pi \in \Pi} f(C,t) \tag{16}$$

It is easy to note that the purpose of cloud computing is to find $f^*(C,1)$ iteratively at the beginning of each horizon and recursively from the end of it. Nevertheless, while Bellman's equation is impractical computationally according to [92], [93], and [21], it identifies the structural properties of the optimal solution.

4.1.3 Model Properties

From a practical point of view, such large space probabilistic dynamic models can be mathematically analyzed with the goal to match particular properties that characterize the optimal solution, describe how argument changes affect the result, and exhibit effective insights into the approximation of the objective solution. In fact, these properties differ depending on homogeneous or heterogeneous VMs involved in the process, namely one class of VMs or multiple classes. The discussion of multiple classes model is deferred to Section 6.1.3.

To cope with this complexity, we will assume that all cloud resources are represented by one class of VMs. In other words, the cloud leases only one class of VMs, which implies the following:

Proposition 1 (Principle of optimality): In-depth analysis of these revenue management problems in the literature shows that the optimal policy exists if $\triangle f(C, t+1)$ is monotone[1] in both capacity and time [97], [56].

Theorem 1 (Monotone property): $\triangle f(C,t)$ is monotone in both capacity C and time t. To confirm that $\triangle f(C,t)$ is decreasing in C, we have to employ some definitions [79].

Definition 1: $c_i^+ \succeq c_i^-$ for $\forall c_i^+, c_i^- \in C_i$ whereas $\succeq$ is a partial ordering relation on the $|N|$ dimensional set $\boldsymbol{C}$.

The intent is to show the relation and order between the elements of the targeted space that is implied by the following definition.

Definition 2: $F(\cdot)$ a real-valued function defined on $|N|$ dimensional set $\boldsymbol{C}$ is partially non-decreasing, supermodular, if $F(c^+) \geq F(c^-)$ and it is partially non-increasing, submodular, if $F(c^-) \geq F(c^+)$ $\forall c^+, c^- \in \boldsymbol{C}$.

For $t = 0$, $\triangle f(C, 0) = 0$, the base case holds corresponding to definitions 1 and 2. Suppose the

[1] The function that preserves the order according to its variables is called monotone function. Particularly, this property denotes to non-decreasing function and concave in the above context [46, 75].

statement is true for $\triangle f(C, t+1)$. Then, we have

$$\begin{aligned}\triangle f(c^+, t) - \triangle f(c^-, t) &= f(c^+, t) - f(c^+ - A^j, t) \\ &- f(c^-, t) + f(c^- - A^j, t).\end{aligned} \tag{17}$$

Assuming the upper condition of the heaviside function (11) is satisfied whenever $j \geq i$, the right-hand side of (12) is maximized. The significance of that is the accepting of any request that offers reward greater than or equal to $a_j R_j$. For the sake of clarity, we refer to those VMs have been released using index l instead of index j. Consequently, the following substitutions are derived from (12):

$$\begin{aligned}f(c^+, t) &= f(c^+, t+1) \\ &+ \sum_{l=1}^{n} q_l(t) \triangle f(c^+ + A^l, t+1) \\ &+ \max_{\vec{u} \in U}\{\sum_{j=1}^{n} p_j(t)(a_j R_j u_j - \triangle f(c^+, t+1)), 0\},\end{aligned} \tag{18}$$

$$\begin{aligned}f(c^+ - A^j, t) &= f(c^+ - A^j, t+1) \\ &+ \sum_{l=1}^{n} q_l(t) \triangle f(c^+ - A^j + A^l, t+1) \\ &+ \max_{\vec{u} \in U}\{\sum_{j=1}^{n} p_j(t)(a_j R_j u_j - \triangle f(c^+ - A^j, t+1)), 0\}.\end{aligned} \tag{19}$$

Similarly,

$$\begin{aligned}f(c^-, t) &= f(c^-, t+1) \\ &+ \sum_{l=1}^{n} q_l(t) \triangle f(c^- + A^l, t+1) \\ &+ \max_{\vec{u} \in U}\{\sum_{j=1}^{n} p_j(t)(a_j R_j u_j - \triangle f(c^-, t+1)), 0\},\end{aligned} \tag{20}$$

and

$$\begin{aligned}f(c^- - A^j, t) &= f(c^- - A^j, t+1) \\ &+ \sum_{l=1}^{n} q_l(t) \triangle f(c^- - A^j + A^l, t+1) \\ &+ \max_{\vec{u} \in U}\{\sum_{j=1}^{n} p_j(t)(a_j R_j u_j - \triangle f(c^- - A^j, t+1)), 0\}.\end{aligned} \tag{21}$$

Likewise, substituting the value of each value function (18), (19), (20), and (21) on the right hand side of (17) produces the following:

$$\begin{aligned}\triangle f(c^+,t) - \triangle f(c^-,t) &= \triangle f(c^+,t+1) - \triangle f(c^-,t+1)\\&+ \sum_{l=1}^{n} q_l(t)(\triangle f(c^+ + A^l,t+1) - \triangle f(c^+ + \triangle A,t+1))\\&- \sum_{l=1}^{n} q_l(t)(\triangle f(c^- + A^l,t+1) - \triangle f(c^- + \triangle A,t+1))\\&+ \sum_{j=i}^{n} p_j(t)(a_jR_j - \triangle f(c^+,t+1))\\&- \sum_{j=i}^{n} p_j(t)(a_jR_j - \triangle f(c^+ - A^j,t+1))\\&- \sum_{j=i}^{n} p_j(t)(a_jR_j - \triangle f(c^-,t+1))\\&+ \sum_{j=i}^{n} p_j(t)(a_jR_j - \triangle f(c^- - A^j,t+1)),\end{aligned} \tag{22}$$

where $\triangle A = A^l - A^j$. Obviously, the right-hand side of (22) is nonpositive; since, $\triangle f(c^+,t+1) - \triangle f(c^-,t+1) \leq 0$ and $\triangle f(c^+,t) - \triangle f(c^-,t) \leq 0$ is followed by induction. On one hand, this property reflects the economic intention to decrease the price when the supplied quantity of VMs to the spot market is high and vice versa. On the other hand, it shows the fact that the cloud provider desires to maximize its expected revenue through increasing quantity supplied in case of abundance of stagnant VMs. The rational behined this is increaing the system utilization through price discrimination. As well, $\triangle f(C,t)$ is decreasing in time. For the sake of clarity let assume that the request is limited to merely one VM. From (13), we have

$$\triangle f(c,t) = f(c,t) - f(c-1,t) \tag{23}$$

then the following substitutions are derived from (12)

$$\begin{aligned}f(c,t) &= f(c,t+1) + q(t)\triangle f(c+A,t+1)\\&+ \max_{\vec{u}\in U}\{p(t)(aRu - \triangle f(c,t+1)),0\},\end{aligned} \tag{24}$$

and

$$
\begin{aligned}
f(c-1,t) &= f(c-1,t+1) \\
&+ q(t) \triangle f(c-1+A,t+1) \\
&+ \max_{\vec{u} \in U}\{p(t)(aRu - \triangle f(c-1,t+1)), 0\}. \qquad (25)
\end{aligned}
$$

From previous proof, we have

$$\triangle f(c,t+1) - \triangle f(c-1,t+1) \leq 0, \qquad (26)$$

$$\triangle f(c+A,t+1) - \triangle f(c-1+A,t+1) \leq 0, \qquad (27)$$

and therefore

$$p(t)(aRu - \triangle f(c,t+1)) \leq p(t)(aRu - \triangle f(c-1,t+1)). \qquad (28)$$

Hence,

$$\max_{\vec{u} \in U}\{p(t)(aRu - \triangle f(c,t+1)), 0\} \leq \max_{\vec{u} \in U}\{p(t)(aRu - \triangle f(c-1,t+1)), 0\}. \qquad (29)$$

Substituting the value of (24), (25), (26), (27), and (29) on the right hand side of (23) results the following:

$$\triangle f(c,t) \geq \triangle f(c,t+1)$$

This is sensible because the chance of rejecting a request is higher at the beginning of a horizon than it is at the end of the horizon. For more information about the non-decreasing and non-increasing properties of a value function, including its representations in this area, we refer the reader to [79], [59] and [57].

Theorem 2 (Concavity property): $f(C,t)$ is concave in C.

Proof: For this purpose, the same approach of [97] can be adopted.

The property in Theorem 1 shows that the total expected revenue increases as the capacity offered to the spot market increases. The property of Theorem 2 shows that the percentage of this increase decreases as the capacity increases. Indeed, these properties have significant importance not merely because they reveal an economic nature, but also have a crucial role in search space reduction for optimization purposes. Especially, when considering the feasible solution of linear programming to solve such a problem as is clear in the following section.

4.2 Dynamic Programming Approximation

Approximate dynamic programming is one of the most widely known approaches used to steer clear of dimensionality. The seminal effort on this context is also credited to Bellman [29]. The primal linear program of such a problem can be written as:

$$\begin{aligned} V \quad &= \quad \max_X \sum_j R_j X_j \\ \text{s.t.} \quad & \sum_j a_{i,j} X_j \leqslant c_i \quad \forall i \\ & X_j \leqslant \sum_t p_j(t) \quad \forall j \\ & \sum_t q_j(t) \leqslant X_j \quad \forall j \\ & X_j \geqslant 0 \quad \forall j \end{aligned} \tag{30}$$

Obviously, to obtain the maximum expected value V of selling X_j expected unit of type j, we have to solve (30) frequently over the horizon. Approximate dynamic programming using linear programming is broadly studied in this area, while [45] proves linear programming provides lower bound of objective values for cost-to-go dynamic formulation, [74] captured upper bound of optimal values.

From this preservative, once the linear formulation is used to capture an optimal solution of (12), the corresponding approximate optimal solution of (10) can be obtained by accepting any request that holds the following inequality:

$R_j + \bar{f}(C - A^j, t+1) \geq \bar{f}(C, t+1)$

where $\bar{f} = V^*$. The outlines of the proposed cloud revenue maximization using the approximate dynamic programming algorithm are given in Algorithm 1 which we refer to as Cloud Revenue

Algorithm 1 CRM-DP

1: Initialization :
2: $C = max.capacity$
3: Set fare types R
4: Set $f(c, T)$ for each $c \in C$
5: **for** all $u \in U$ **do**
6: **for** all $c \in C$ **do**
7: Set one-step probability transition matrix P_{uc}
8: **end for**
9: **end for**
10: $t = T - 1$
11: **while** $t \geq 0$ **do**
12: Compute:
13: $\bar{f}(C, t+1)$
14: $\bar{f}(C - A^j, t+1)$
15: **if** $(R_j + \bar{f}(C - A^j, t+1) \geq \bar{f}(C, t+1))$ **then**
16: $u = 1$ "Accept"
17: **else**
18: $u = 0$ "Reject"
19: **end if**
20: $t = t - 1$
21: **end while**

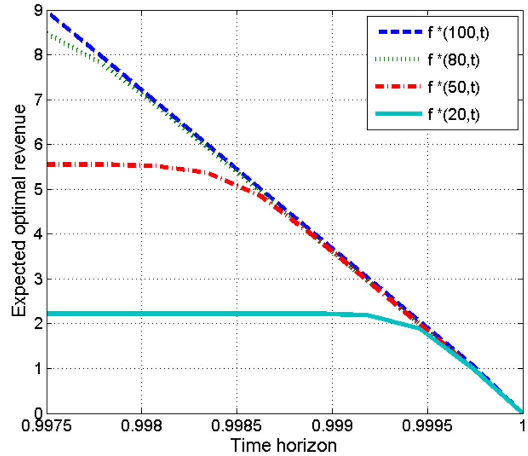
Maximization-Dynamic Pricing (CRM-DP). Noteworthy that lines 5 to 9 can be updated every time epoch to satisfy the heterogeneous nature of demands.

4.3 Performance Evaluation

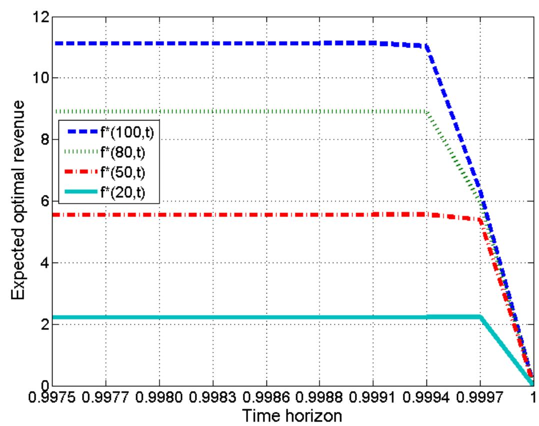
In this section, we present detailed numerical results of optimal revenue to validate the optimal policy and value function properties. We provide expected optimal revenue comparisons through soft and robust demand rates, using multiple capacities. We discuss the mutual effect of horizon length and demand rate on the optimal revenue. For both soft and robust rates, we study the dynamics of price over the horizon. Then, we present the running time of our formulation compared with Hong's approach [97].

4.3.1 Impact of Demand Rate

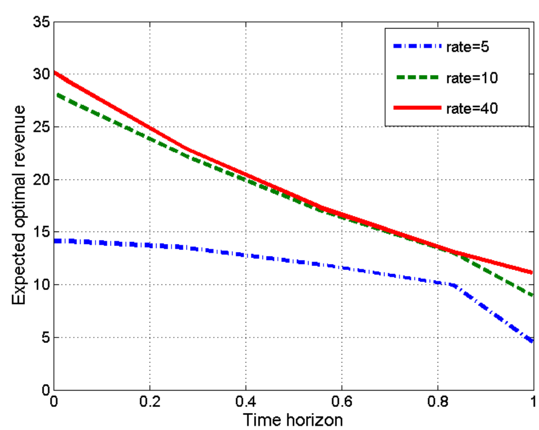
The maximum allowed candidate capacity of the system for plug in to the dynamic pricing scheme C is set to 100 VM. The time horizon and price are normalized to $[0, 1]$, where the one hour decision horizon length is divided into 60×60 intervals; each decision interval t is 2.7×10^{-4}. Figs. 4.3.1a and 4.3.1b present the expected optimal revenue within the last 10 slots of variant capacities. We observe that the optimal revenue converges to the value at the beginning of the horizon and monotonically decreases close to the closing time, which satisfies model properties in Section 4.1.3. In case of soft arrival demands ($\gamma = 10$), the optimal revenue rapidly decreases in time, particularly in a plethora of capacity ($C = 100$), as shown in Fig. 4.3.1a. Moreover, Fig. 4.3.1b reveals that a very high demand has a minor effect on short remaining time ($f^*_{\gamma=10}(100, 0.9975) = 8.96$ versus $f^*_{\gamma=60}(100, 0.9975) = 11.13$). Fig. 4.3.1c shows how the time has a weak effect on soft arrival rate ($\gamma = 5$), apart from closing time, as compared with a strong effect. In the case of robust arrival rate ($\gamma = 40$) over the entire horizon, this is reasonable because it reflects the natural phenomenon of more time, more gain. Thus, the results of Fig. 4.3.1 correlate with the monotone property of the value function in Section 4.1.3.



(a) Mean rate $\gamma = 10$



(b) Mean rate $\gamma = 60$

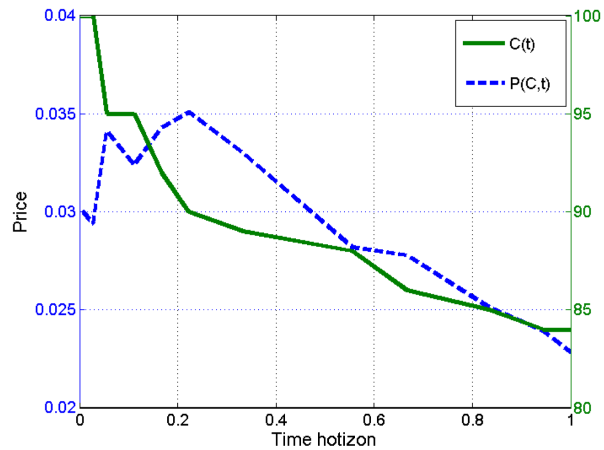


(c) Revenue vs. rates

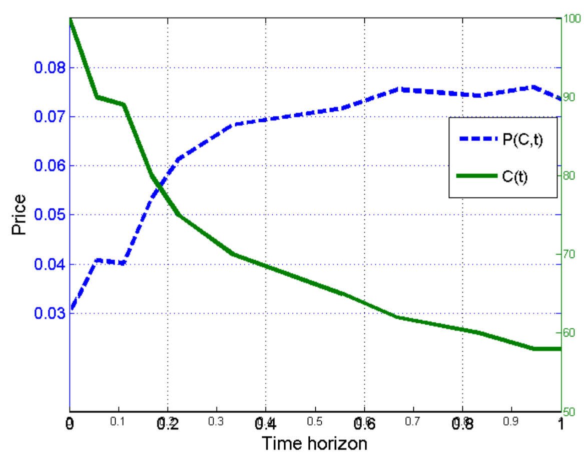
Figure 4.3.1: Expected optimal revenue

4.3.2 Optimal Price

Fig. 4.3.2 depicts the results for two different scenarios. When $\gamma = 5$ (low arrival rate), the optimal price decreases with a slight consumption of capacity, indicating the attraction of more customers. Further, a considerable decrease in capacity leads to a rapid increase in price, which indicates attempts to increase the expected revenue. This is clearly shown in intervals $[0, 0.2]$ of Fig. 4.3.2a. In the other scenario, when $\gamma = 60$, it is easy to note that the optimal price and remaining capacity are inversely related. Thus, Fig. 4.3.2b implies that a high demand rate and capacity consumption increase the optimal price, especially at open time. So far, we have focused on optimal expected revenue and price properties. We will now study the optimal policy of our algorithm.



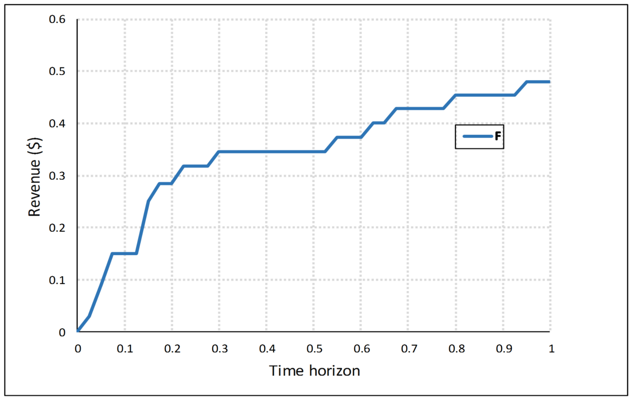
(a) Low mean rate $\gamma = 5$



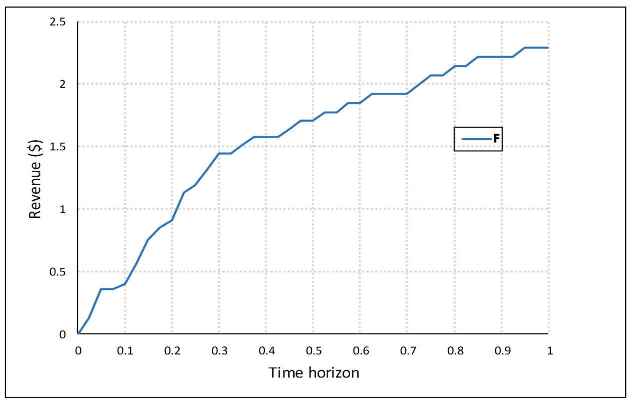
(b) High mean rate $\gamma = 60$

Figure 4.3.2: Price and running time

Fig. 4.3.3 depicts the revenue gained from the two different scenarios listed above. When $\gamma = 5$ (low arrival rate), the earned revenue at the end oh horizon is \$0.479. While, when $\gamma = 60$ (high arrival rate), the earned revenue is \$2.293.



(a) Low mean rate $\gamma = 5$



(b) High mean rate $\gamma = 60$

Figure 4.3.3: Revenue and running time

4.3.3 Optimal Policy

Recalling from Section 4.1.2, with a view to achieving maximum expected revenue over the horizon, we have to select the optimal decision u^* (15) that leads to the optimal policy π^*. The corresponding numerical results are presented in Fig. 4.3.4. Here, we assume that the total capacity $C = 20$, with two different demand rates. Fig. 4.3.4a shows that the optimal policy π^* is $\{0, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$ where the remaining VMs is 9 and the demand rate $(\gamma = 5)$ is low. By contrast, Fig. 4.3.4b shows that it is $\{0, 0, 0, 1, 1, 1, 1, 1, 1, 1\}$ when the demand rate $(\gamma = 10)$ is higher. This means that it is better to reject the request when there is enough time to sell, in case of robust demand, to aim at higher profit in the future.

Available capacity \ Stage	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1

(a) Optimal Policy with $\gamma = 5$

Available capacity \ Stage	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	0	1	1	1	1	1	1
8	0	0	0	1	1	1	1	1	1	1
9	0	0	0	1	1	1	1	1	1	1
10	0	0	0	1	1	1	1	1	1	1
11	0	0	1	1	1	1	1	1	1	1
12	0	0	1	1	1	1	1	1	1	1
13	0	0	1	1	1	1	1	1	1	1
14	0	1	1	1	1	1	1	1	1	1
15	0	1	1	1	1	1	1	1	1	1
16	0	1	1	1	1	1	1	1	1	1
17	0	1	1	1	1	1	1	1	1	1
18	0	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1

(b) Optimal policy with $\gamma = 10$

Figure 4.3.4: Optimal policy with entire capacity $C = 20$ and variant rates

4.4 Conclusions

The use of thoughtful spot price of virtual machines has a significant role in contemporary cloud economy because it not only directly relates to their revenue, but also mitigates cloud congestion. Both the uncertainty of arrival and departure demands and high state complexity pose a challenge to address revenue maximization in real time. Therefore, we proposed a formulation of profit maximization, using a discrete finite horizon, characterization of optimal properties and controlling conditions, and an approximation of dynamic processes using linear programming. Unlike most previous methods of maximizing cloud profitability, which focus on feasibility, our CRM-DP shows practical and new dynamic analysis insights. Additionally, the numerical results capture and define the optimal policy and demonstrate our analysis. This work amounts to a step ahead in IaaS cloud revenue maximization. It also opens the door to many ideas for future work such as creating more efficient dynamic algorithms to involve different classes of VMs in Chapter 6.

Chapter 5

Dynamic pricing: A Column Generation Approach

The limited works on dynamic pricing consider selling the entire population of VMs in the cloud, resulting in increased processing time and risk. Stagnant resource might produce a significant number of VMs that must be supplied to the spot market to maximize cloud revenue. In this chapter, we seek to expose the reasons behind the requirement to use dynamic pricing schemes in IaaS cloud. In addition, we are looking at finding an approach to mitigate time complexity of our previous model for practical purposes.

5.1 Proposed Pricing Model

The key role of dynamic pricing is to automatically adjust prices in response to the law of supply and demand, in order to maximize both utilization levels and income. However, most discussions regarding Amazon's spot pricing demonstrate that it is artificially updated [22,51,97]. Spot instances' pricing history exhibits an unreasonable spike in price for the m2.2xlarge instance in September of 2011 [15,49]. Motivated by the aforementioned issues as well as the success stories of dynamic pricing that have been achieved in widespread applications (e.g. airline and hotel reservations) [80], this chapter seeks to answer the following questions:

- Why involve dynamic pricing in cloud computing?
- What type of stochastic process is involved?
- Which factors affect the dynamic pricing of cloud computing?
- How can demands be mapped to stagnant VMs?
- Is it possible to find the optimal policy that promotes the highest revenue in the right amount of time?

The requirements to respond to these questions underline the insights and hence methods, that may be effective in mitigating the problem. Using the general framework provided by the aforementioned questions, this chapter formulates a column generation (CG) approach to linear relaxation of the maximum expected revenue, under discrete-time stochastic Markov decision processes over a finite horizon.

5.1.1 Dynamic price: Objectives and Necessities

Answering the first question uncovers the necessities and objectives that drive and encourage cloud providers. The adoption of a dynamic pricing strategy in the cloud by any organization emphasizes its necessity. For instance, Amazon states that the customer bids the maximum price they are willing to pay per instance per hour on Amazon's spare EC2 instances, and the request is accepted if the bid price exceeds the current Spot price [3]. The Spot price, assigned by Amazon, fluctuates such that the price decreases to run the service, or it increases to terminate the service when the demand for the instance increases, or when the instance's supply decreases. On the other hand, Google presents a finite set of virtual machines for sale in its Preemptible VM Instance, which is only available during non-peak periods and terminates the running instances when the underlying resources are needed [12]. Furthermore, Google deems that the price roughly follows Moore's Law [13]. It is worth noting that both charge the customer a fixed price per hour during one round, from the beginning until termination. It is evident that they offer a set of VMs for a limited time and at a volatile price, depending on certain conditions, despite seemingly providing a cost-saving service. The question remains as to how these VMs are generated and how they are terminated. Changes in price are an implicit consequence of demand for VMs in reserved and on-demand pricing models. From a technical point of view, virtualizing a physical machine (PM) leads to a limited set of VMs that vary in type and quantity depending on the supply (i.e. the type of available PM), in addition to the demand (i.e. the requested VMs). For example, assume that there is an on-demand or reserved request with targets of type A VM. The cloud accepts this request by operating a PM, which could be scaled up to four VMs of the desired type A. In reality, the cloud loses the potential profit of such machines (taking into account the power consumption and other operational costs), as well as the VMs that were prepared for the sake of high availability and disaster recovery. We argue that a dynamic pricing model, without a service level agreement (SLA), could be the best pricing model for utilizing such virtual machines (called stagnant VMs). For the customer's perspective, it could attract customers whose jobs have batch or interrupt-tolerant features if it meets their demands (i.e. type, quantity, and price). It is clear that utilizing redundant and randomized sets of stagnant VMs in order to maximize cloud marginal revenue are not trivial objectives, considering the randomness involved in many aspects.

5.1.2 Dynamic price: 3D Stochastic Property

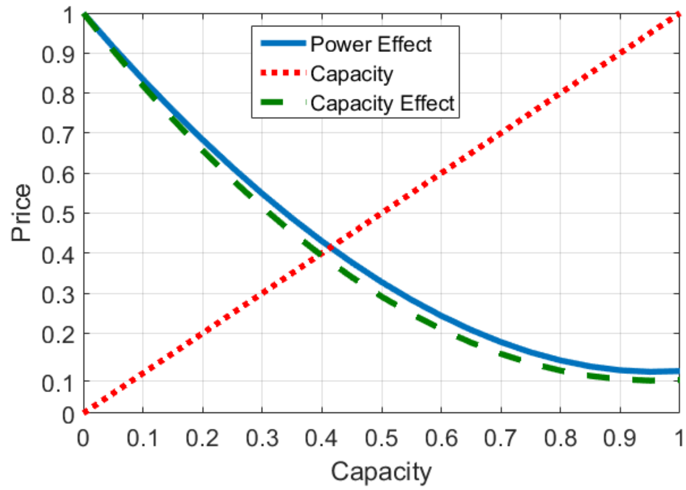
Initial analysis of tackling such issues addresses stiff challenges, specifically the three-dimensional ($3D$) stochastic dilemma. First, uncertainty in arrival rate or purchase demand has been noted in other domains, such as airline seats and hotel booking. Second, efforts made to resolve this problem add yet another dimension of uncertainty, namely the departure rate or demand to leave. Third, the diversity of cloud states increases when there are many VMs, as the cloud fluctuates between active and stagnant states. In fact, the $3D$ triple stochastic axis dramatically increases the size of the search space for finding the optimal solution. This provides interesting incentives and complex analytical insights. It is worth noting that many researchers [80, 92] in an analogous research domain have utilized the Poisson probability distribution, since it is the most convenient distribution to describe the uncertainty of arrival and departure processes. Furthermore, one of the critical issues that emerges from applying the dynamic pricing approach in an IaaS cloud environment is response time. This imposes an immediate decision, which disappears in other applications where the provider has enough time to accept or reject the request. The following sections outline our approach to overcoming such obstacles after establishing the dynamism mechanism adopted.

5.1.3 Dynamism Parameters

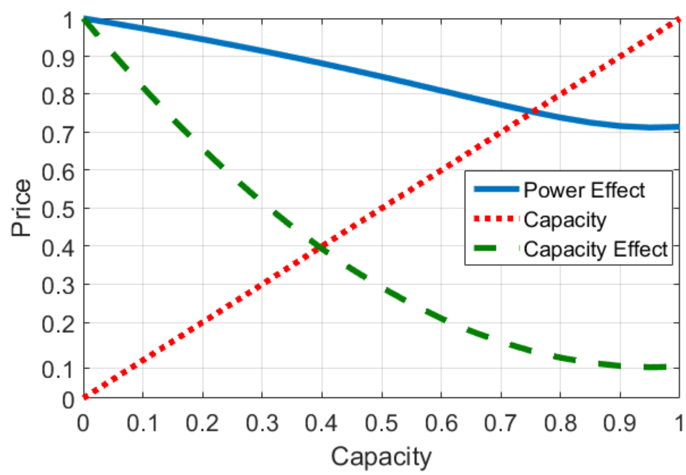
In this section, we strive to define the key factors that have a vital role in pricing dynamism and hence formulate a price in terms of the dominant parameters. To achieve this goal, we capture and quantify the potential implications of parameters employed by various authors [97, 99, 101]. Consequently, we observe that capacity, the number of available VMs, and time (i.e. the moment of sale) are the most prominent parameters. These observations reflect the economic point of view that, on one hand, the quantity of states increases as the price decreases, and vice versa; on the other hand, the more time available to sell, the more flexible opportunity there is to gain [64]. Additionally, one should not lose sight of the actual cost of VMs, which relies on numerous overlapping factors and resources beyond the scope of this thesis. However, one can observe the effect of power consumption costs on total cloud revenue [58] and, more specifically, the correlation between power consumption and virtualization [55, 60]. The latter shows how power consumption is directly proportional to virtualization processes. This insight inspires authors to leverage a periodic power consumption variance as an active player in the dynamic pricing mechanism. Another consequent advantage arises from employing power consumption as an automatic and scalable meter for the purpose of mitigating congestion. To the best of our knowledge, this is the first work to leverage power consumption in creating a dynamic pricing model aimed at maximizing cloud revenue. To achieve this goal, let $\rho(t) \in [0, \rho_{max}]$ denote the price of a given VM at time t, where $\rho_{max} \leq 1$. We define the price as a function of power consumption w and capacity c at time t as shown in equation 31:

$$\rho(t) = ((1 - c(t))^a + \epsilon)^{e^{-w(t)b}} \tag{31}$$

where a, b and ϵ are calibration parameters related to the cloud state. Such a definition appears convenient in our case since two discriminating power consumption levels work as sentinel values. Fig. 5.1.1 depicts the effect of low power consumption, namely low utilization, and high power consumption, namely high utilization, on the behavior of price in terms of available capacity and time.



(a) Low Power Consumption



(b) High Power Consumption

Figure 5.1.1: Price vs. Capacity and Power

The former, Fig. 5.1.1a, shows that low power consumption implies that the available capacity will become a dominant dynamic parameter in price differentials, where the price $\rho(t) \simeq (1-c(t))^a + \epsilon$ when $wb \simeq 0$. The empty set of available VMs, $c(t) = 0$, allows the price to increase to the maximum value, as an implicit condition that imposes a cloud price of $\rho(t) = \rho_{max} \simeq 1$ when the stock is zero. In contrast, the cloud forces the price to be zero, $\rho(t) \simeq 0$, in the case of redundant capacity $c(t) \simeq 1$. The latter, Fig.5.1.1b, reveals how high power consumption significantly affects the price by constricting it to the maximum value as a binding condition, in order to mitigate cloud congestion. This occurs when $wb \simeq 1$.

Thus far, our focus has remained on endogenous factors. We now move to discuss the crucial exogenous factors. We adopt the same notions of arrival and departure rate as [23, 24, 97]. We can loosely assume that demands, namely arrival and departure rates, are functions of price at a certain time and that the acceptance or rejection processes do not affect these rates. Furthermore, to model economic perspectives, we assume that:

- $\lambda_t(\rho)$, the demand to hold a VM of price ρ at time t, is non-increasing in ρ as well as $\mu_t(\rho)$, the demand to release a VM of price ρ at time t, is non-decreasing in ρ.
- when price descends to the lowest level $\rho(t) = 0$, the demand to hold the VM ascends to the maximum expected value $\lambda_t(0) = \gamma$; as a result, theoretically there is no incentive to leave the system, $\mu_t(0) = 0$.
- Conversely, when price ascends to the maximum level $\rho(t) = 1$, there is no incentive to hold a new VM $\lambda_t(1) \simeq 0$, thus forcing customers to leave cloud services $\mu_t(1) \simeq \beta$.
- The mean arrival rate γ and mean departure rate β are non-negative.

Fig. 5.1.2 shows an example of demand curves derived from the following formulas

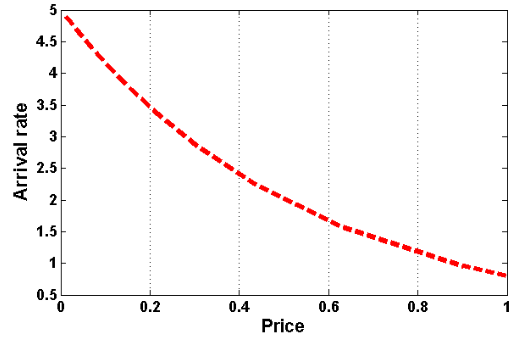
$$\lambda_t(\rho) = \gamma / e^{\alpha\rho(t)} \text{ and} \tag{32}$$

$$\mu_t(\rho) = \beta(1 - 1/e^{\alpha\rho(t)}), \tag{33}$$

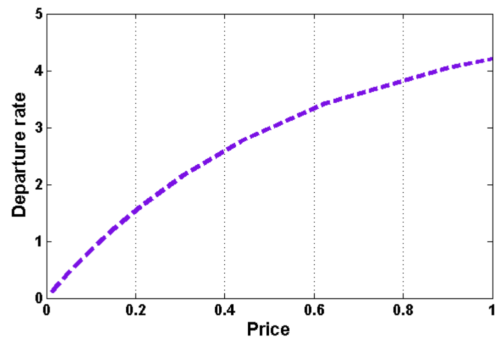
where the scaling parameter $\alpha \geq 1$.

5.1.4 Optimum Expected Revenue

Using this general framework, we create and formulate an infrastructure as a service (IaaS) cloud model that presents and solves the problem, finding the optimum mapping from stochastic demands to available stagnant virtual machines and hence addressing it with the view of maximizing marginal



(a) Arrival rate



(b) Departure rate

Figure 5.1.2: Arrival and departure rates ($\gamma = 5$ and $\beta = 5$)

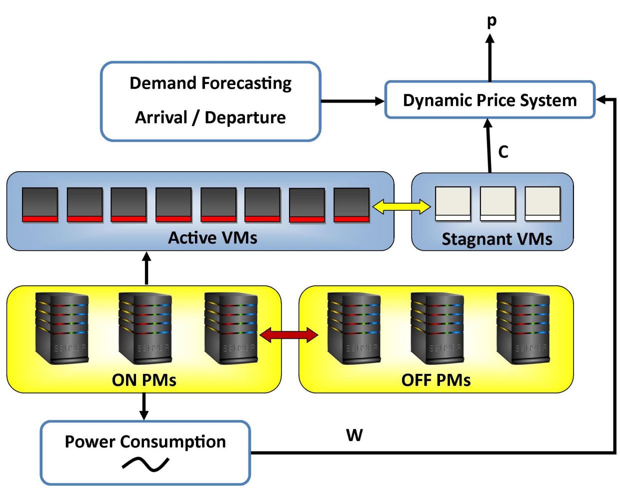


Figure 5.1.3: A system model.

cloud revenue. To fulfill these purposes, Fig. 5.1.3 presents a state of an IaaS cloud and assumes that a pool of $|N|$ stagnant VMs are available for sale under a dynamic pricing model. It is obvious that price discrimination varies based on any change in power, demand, or resources.

The state of the system at time t is $C \in \mathbf{C} = \{0, 1, ...|N|\}$, and the expected profit of the current state must be considered within a finite horizon H, which is divided into T monitoring epochs Δt, where $\Delta t = 1$ and $\Delta t = T$ represent the initial and final epochs, respectively. Poisson processes require that each epoch must be small enough to allow handling only one request. However, the request includes not only one unit of instance, but most likely a set of VMs. Each set $A \in \{0, 1, ..., |N|\}$ represents a type of service and directly contributes R_A fare to the cloud's revenue.

The problem concerning dynamic pricing strategies can be formulated as a finite horizon Markov decision process under uncertainty. We consider a discrete time horizon in which, at any given interval and any possible state of the Markov chain, the system selects the optimal decision, to sell or to wait, by traversing all stochastic possibilities of subsequent rewards until the end of the horizon, with a view to maximizing marginal cloud revenue. A request of type A, demanded by the customer, launches the cloud market at the beginning of the horizon along with a finite space C of stagnant VMs supplied by the cloud for sale. The dynamic price system either accepts or rejects the request, according to the results of evaluating the expected profit of demanded resources until closing time, the end of the horizon. Approval transfers the cloud to state $(C - A, t + 1)$ in the next epoch, raising cloud revenue by R_A. Refusal maintains the cloud state at $(C, t + 1)$ without additional revenue. On the other hand, leaving the service changes the space of the stagnant pool by adding $\bar{A}$ VMs. Thus, if we assume that at any time interval (t), the demand to hold the VM occurs with probability $p(t)$ and the demand to release the VM occurs with probability $q(t)$, the system must take the decision u_A according to the current state C and the customer's bid R_A, aiming to maximize the total expected revenue $f(C, t)$. This could be evaluated using Hamilton-Jacobi-Bellman's equation, as shown in equation 34:

$$
\begin{aligned}
f(C,t) = \max_{\vec{u}\in U}\{\sum_{j=1}^{n} p_j(t)(R_A u_A + f(C - Au_A, t+1)) \\
+ \sum_{j=1}^{n} q_j(t) f(C + \bar{A}, t+1) \\
+ (1 - (\sum_{j=1}^{n} p_j(t) + \sum_{j=1}^{n} q_j(t))) f(C, t+1)\},
\end{aligned}
\tag{34}
$$

subject to the terminal conditions:

$$
\begin{aligned}
& f_u(C,0) \triangleq 0, \\
& f_u(C,T+1) \triangleq 0, \text{ and} \\
& f_u(0,t) \triangleq 0, \quad \forall t,\ \forall C
\end{aligned}
$$

This evaluation can be used because the revenue must be set to zero before launching the market ($t = 0$), after closing the market ($t = T + 1$), and when the pool of stagnant VMs becomes empty ($C = 0$). It is worth noting that to fulfill the ultimate objective, the dynamic price system computes $f^*(C,1)$. For more detailed information regarding discrete finite horizon stochastic Markov decision processes, the properties of model formulation (34), and the optimal policy of maximizing revenue, we refer the reader to other research [23, 97].

As mentioned in section 5.1.2, solving (34) in order to find the optimal decision policy u^* suffers from a triple stochastic space, rendering the problem exhausting and impractical. Consequently, relaxing dynamic programming is a substantial prerequisite for overcoming the imposed high dimensionality.

5.2 Column Generation Pricing Model

One of the interesting approaches that may be attempted in order to surpass the dimensionality barrier is to approximate dynamic programming based on linear programming (LP). We thus reformulate (34) to maximize the expected objective function V, by expecting to sell X_j units of VM using fare R_j within the horizon as follows:

$$
\begin{aligned}
V \quad = \quad & \max_X \sum_j R_j X_j \\
& \text{s.t.} \ \sum_j a_j X_j \leqslant C \\
& \quad X_j \leqslant \sum_t p_j(t) \quad \forall j && (35) \\
& \quad \sum_t q_j(t) \leqslant X_j \quad \forall j && (36) \\
& \quad X_j \geqslant 0 \quad \forall j
\end{aligned}
$$

where a_j represents number of VMs in unit X_j, and (35) and (36) are arrival and departure constraints, respectively.

It is worth noting that a degree of high complexity arises from the constraints. We employ column generation to address such a large-scale problem and adopt the Dantzig-Wolfe decomposition approach [39].

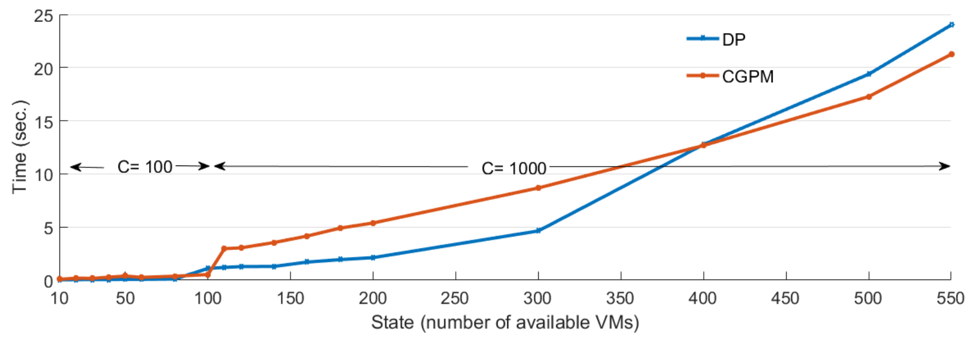
A column generation approach is especially efficient for solving linear programming problems that sustain large packages of constraints [62]. Gilmore and Gomory proved this efficiency through their successful practical application when solving the cutting stock problem [43]. This technique works quite well with the revised simplex method, which has a sparse and structured constraint matrix, since the problem could be decomposed into two sub-problems, a master LP problem and an auxiliary LP problem (called the pricing problem). The former's constraints matrix involves a huge number of columns. The explicit enumeration of all columns reach for the optimal solution, but such a process is considerably expensive. The subtle idea proposed by Dantzig and Wolfe states that only a subset of these columns is required to get the solution, because the associated variables of the vast majority of the columns coincide with zeroes in the optimum solution. The later, the pricing problem, is responsible for generating a new, promising column that improves the objective function of the master LP or determining the optimal solution. The new column is added to the master problem, which produces the dual prices (with non-negative reduced cost coefficient). Then the dual solution is passed to the pricing problem. The iterative interactions between master and pricing problem terminate when no new column is generated with a non-negative reduced cost. It is worth mentioning that column generation could be accelerated when the problem is within distance-K tolerance of the optimal solution by terminating the process before generating the rest of the columns that cover the distance K between the current solution (obviously a near optimal solution) and the optimal solution. To fulfill this, a certain number of iterations or a reduced cost criterion could be used. For more information about column generation and Dantzig-Wolfe decomposition, we refer the reader to [38].

5.3 Performance Evaluation

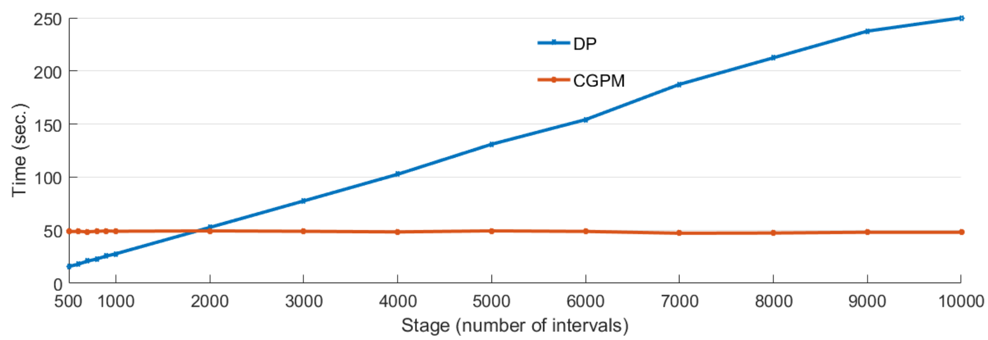
This section records the empirical evaluation results. We involve a column generation approach to solving the linear relaxation of the dynamic programming model. The purpose of this evaluation is to show how well our spot pricing model compares to other spot pricing models. All our empirical evaluations are performed by Matlab version R2015b on an Intel Core i7 machine at 3.6 GHz with 32 GB RAM.

5.3.1 Numerical Results

We consider a discrete time finite horizon Markovian decision in which the system maximizes revenue by selecting the right decision within a certain time. The number of states, namely the capacity of stagnant VM candidates for sale, and the length of horizon are radical criteria for evaluating runtime (or response time). Therefore, we begin by studying runtime for the column generation pricing model (CGPM) and the standard stochastic dynamic programming model (DP). For this purpose, we assume the time interval to be 1 second and the length of horizon to be 3600 intervals. Fig. 5.3.1a depicts runtime versus number of available VMs for sale. We discriminate two scenarios; the maximum capacity allowed for spot market is C=100 and C=1000. It is clear that the runtimes of



(a) Time versus capacity



(b) Time versus stages

Figure 5.3.1: Price versus capacity and stages

DP and CGPM are very close when the state is less than 100, whereas there is a clear superiority for DP when the system state varies between 100 and 400 VMs. However, CGPM beats DP if the current state is selling more than 400 VMS (supplying a huge number of VMS). To see the effect of the length of horizon on the performance, we evaluate the runtime over the course of horizons spanning from 500 upto 10000 intervals. Fig. 5.3.1b shows that CGPM outperforms DP when the length of horizon exceeds 2000 intervals. The runtime of CGPM shows stability versus the number of stages while DP grows rapidly with it. Moreover, Figs. 5.3.2 and 5.3.3 depict the gap (error) between the optimal expected solution and the solution obtained by CGPM. Note that in 26% of cases, the solutions are identical (0 gap) and in 43%, the gap is less than 0.5. This indicates that approximately 69% of gaps are less than 0.5. Indeed, this varies depending on the adopted stopping criteria and cutting point used to create the restricted master problem.

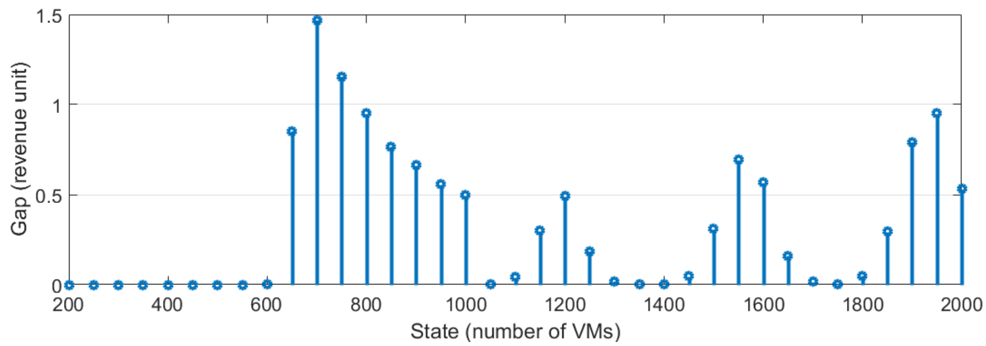


Figure 5.3.2: Gap from optimum solution

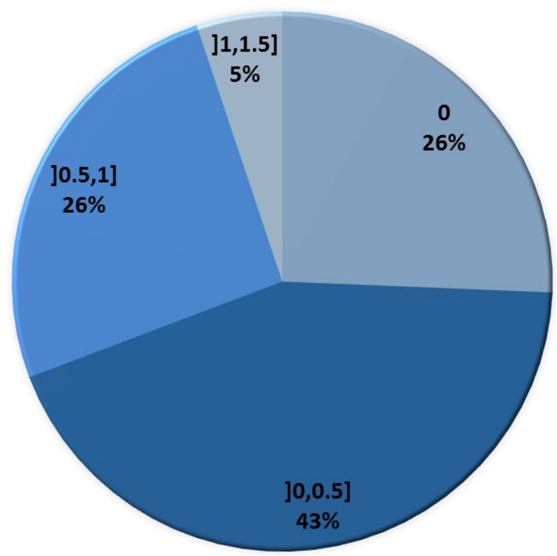


Figure 5.3.3: Gap percentage

5.3.2 Case Study

This case study refers to the pricing strategy of Amazon's Spot instances. To confirm the performance of our model, we study four cases (A-D), as shown in Fig. 5.3.4. In all cases, the horizon is divided into 10 intervals; the number above the vertical arrow represents the number of requested VMs, the number below the vertical arrow is the customer bid for each VM, and the maximum capacity of VMs supplied for spot market $C = 5$. Moreover, we assume that at the beginning of the horizon, two VMs have been rented for \$0.50 each. For instance, case A assumes two requests. The first calls for one VM with \$0.40 at time interval 4 while the second also bids \$0.40 for one VM but at interval 8. There are $c(t = 0) = 5 - 2 = 3$ non-utilized VMs at the beginning of the horizon. Table 5.1 tabulates our findings. The results show that our CGPM model does better than Amazon's strategy [11] in terms of profit, even when utilizing a number of VMs less than Amazon, as in case B. The rationale behind this is that our model rejects the first bid ($0.30), because it is considered as a worthless bid with high mean arrival rate $\gamma = 4$, and then accepts the consecutive bids.

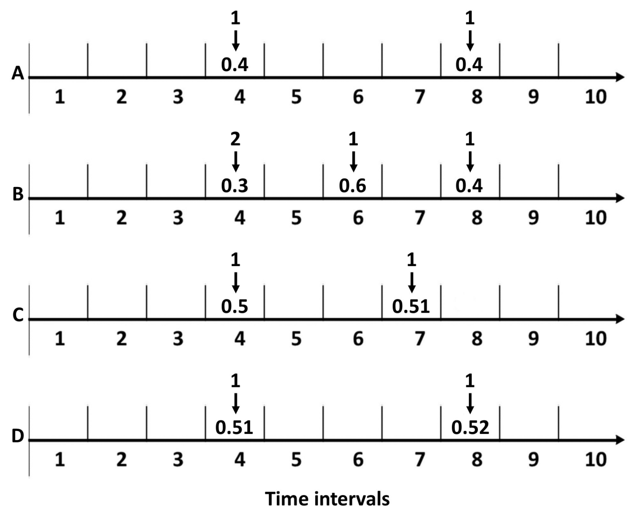


Figure 5.3.4: Bids case study: Five VMs available for sale within ten intervals

Case	Profit		Utilization		γ	c(t=0)
	Amazon	CGPM	Amazon	CGPM		
A	12	13.2	4	4	2	3
B	12.4	13.2	5	4	4	3
C	14.5	14.53	4	4	2	3
D	25	25.1	5	5	2	0

Table 5.1: Profit and utilization comparisons

5.4 Conclusion

This chapter proposed a column generation approach to a relaxed dynamic pricing model. The aim is to provide cloud organizations with the ability to utilize spare VMs and hence to maximize cloud revenue. We characterized dynamic pricing necessities, objectives, and parameters. According to those, we formulated the maximum expected revenue using the column generation technique and stochastic Markov processes over a finite horizon. We employed our methodology for studying the performance of cloud spot pricing pioneers. Our proposed approach showed that it is feasible for large-scale Spot instances.

Chapter 6

Multiple classes dynamic price scheme

In this chapter, we discuss the design and implementation of a multiple class dynamic pricing approach and infer its effects on the revenue. In order to facilitate the basic understanding of our approach, we primarily recall some basic concepts related to the mechanism of dynamic pricing scheme dedicated for one class of VMs.

6.1 System Model

6.1.1 Modeling Assumptions

In particular, the objective of this chapter is to extend the previous model to include multiple class of VMs. That is to determine the best utilization of stagnant VMs to maximize the revenue of the cloud through spot market.

For the sake of clarity, before delving into our approach details, we will use the notations of Table 6.1 over the course of this chapter. Let a cloud provider have $|N|$ class of VMs where the available classes in the data center is $n \in N = \{1, 2, ..., |N|\}$. The state space C is a set that represents the number of instances of VMs, where $c_j(t) = k$ represents k instance of class $j \in \{1, 2, ..., n\}$ at time t, $c_j(t) \in K \triangleq \{0, 1, ..., k\}$. Based on these notations, the state of the system, the pool of stagnant VMs resources of all classes which are available for sale, at any given interval t is denoted as $C(t) = \{c_1(t), c_2(t), ..., c_n(t)\}$. Practically, the demand includes not merely one class of VMs but perhaps a variety of classes. Each class j of virtual machines directly contributes

Table 6.1: Key notations

Notation	Description
$\|N\|$	number of VMs classes.
j *and* l	class indices.
T	time length of sale horizon H.
$\triangle t$	shortest time interval in which only one incident occur.
$c_j(t)$	number of instances of class j available at time t.
$C(t)$	set of all VMs of all classes available for sale at time t. $C(t) = \{c_1(t), c_2(t), \ldots, c_n(t)\}$ in case of n class, otherwise $C(t) = \{c\}$.
a_j	number of demanded VMs of class j.
$R_j(t)$	reward induced from selling one unit of class j at time t.
R	total reward at the end of horizon.
$\rho_j(t)$	price for one unit of class j at time t.
ρ_{max}	maximum price assigned for VM
$\lambda_j(\rho, t)$	arrival rate for class j of price ρ at time t.
$\mu_j(\rho, t)$	departure rate for class j of price ρ at time t.
γ	mean arrival rate in the horizon.
β	mean departure rate in the horizon.
A	request of n class set of VMs.
A^j	request for lease or release amount of class j VMs.
$p_j(t)$	probability of request to hold VM of class j at time t.
$q_j(t)$	probability of releasing VM of class j at time t.
$f(C, t)$	total expected revenue of available resources C at time t.
$\bar{f}(C, t)$	approximate maximum revenue of resources C at time t.
u_j	decision to accept or reject a request for class j.
U	the set of decisions.
π^*	optimal n-vector policy.
P_u	3-dimensional one-step probability transition matrix.
V^*	the optimal expected revenue using linear programming.

a reward of $a_j R_j(t)$ to the cloud revenue, where a_j is the quantity of VMs demanded of class j. Then the total reward

$$\mathbf{R} = \sum_{t=1}^{T} \sum_{j=1}^{n} a_j R_j(t),$$

and without a loss of generality let order classes rewards $R_1(t) \leq R_2(t) \leq ... \leq R_n(t)$. Assume the time horizon H, the length of horizon T, the price $\rho_j(t)$ of class j at time t, the arrival rate $\lambda_j(\rho, t)$, and the departure rate $\mu_j(\rho, t)$ are defined and characterized as in Section 4.1.1.

6.1.2 Model Formulation

The problem of dynamic pricing strategy for multiple classes can be formulated as a finite horizon Markov decision process under uncertainty.

At any given interval $\triangle t$, for clarity denoted as t, non-investing capacities of resources, represent a finite space C of cardinality $|C| = n$ supplied by cloud for sale, is represented as (C, t) and an n-class request $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$, demanded by the customer. If the system accepts the request of class j, i.e., $A^j = \{0, \ldots, a_j, \ldots, 0\}$, it moves to the state $(C - A^j, t+1)$ in the next epoch. In case the system rejects the request, the state becomes $(C, t+1)$. On the other hand, the system moves to the state $(C + A^j, t+1)$ as a result of releasing class j VM from the customer. At each time interval, Poisson incurs only one request of arrival or departure, if any, with that being a request to hold a class j, which arrives in time t with probability $p_j(t)$, or to release it with probability $q_j(t)$. Instantly, a binary decision u_j, to sell or to wait, must be taken according to the current state. In case of a sale, class j customer offers reward of $a_j R_j(t)$. It is worth noting that in reality two or more requests can happen at a time. This case could be circumvented by considering one request and postponing the rest to the subsequent time intervals. Especially since the time interval is very small.

From the aforementioned, we can formulate the total expected revenue $f(C, t)$ as shown in equation 37:

$$\begin{aligned} f(C,t) &= \max_{\vec{u} \in U} \{ \sum_{j=1}^{n} p_j(t)(a_j R_j u_j + f(C - A^j, t+1)) \\ &+ \sum_{j=1}^{n} q_j(t) f(C + A^j, t+1) \\ &+ (1 - (\sum_{j=1}^{n} p_j(t) + \sum_{j=1}^{n} q_j(t))) f(C, t+1) \}, \quad \forall t, \ \forall C, \end{aligned} \tag{37}$$

subject to the terminal conditions:

$$
\begin{aligned}
&f(C,0) \triangleq 0,\\
&f(C,T+1) \triangleq 0,\ and\\
&f(0,t) \triangleq 0, \qquad \forall t,\ \forall C.
\end{aligned}
$$

Clearly, the maximization process implies a decision to satisfy

$$
u_j = \begin{cases} 1 & if\ a_j R_j(t) + f(C - A^j, t+1) \geq f(C, t+1) \\ 0 & otherwise, \end{cases} \tag{38}
$$

to select the promising action, either to sell, incrementing the reward by $a_j R_j$, or to wait. The value of the objective function (37) from a given state can be briefly formulated to

$$
\begin{aligned}
f(C,t) &= f(C,t+1) + \sum_{j=1}^{n} q_j(t) \triangle f(C + A^j, t+1) \\
&+ \max_{\vec{u} \in U} \{ \sum_{j=1}^{n} p_j(t)(a_j R_j u_j - \triangle f(C, t+1)), 0 \}, \forall t,\ \forall C,
\end{aligned} \tag{39}
$$

To achieve the cloud's objective (39), namely to maximize the total expected rewards, the system finds

$$
u_j^* \in \underset{\vec{u} \in U}{\operatorname{argmax}} \{ \sum_{j=1}^{n} p_j(t)(a_j R_j(t) u_j - \triangle f(C, t+1)) \}. \tag{40}
$$

6.1.3 Model Properties

Up to this point we have an idea of the properties of one class model, where all VMs are identical. The theorems in Section 4.1.3 are true under a strict condition. All cloud resources are represented only by one class of VMs, which is admittedly a large assumption that requires further investigation. Consequently, we have to propose properties of multiple class model. In other words, the customer is able to request a variety of VMs simultaneously (i.e. one VM of the first class, zero VM of the second class, three VMs of the third class, and so on). Each VM class has a particular

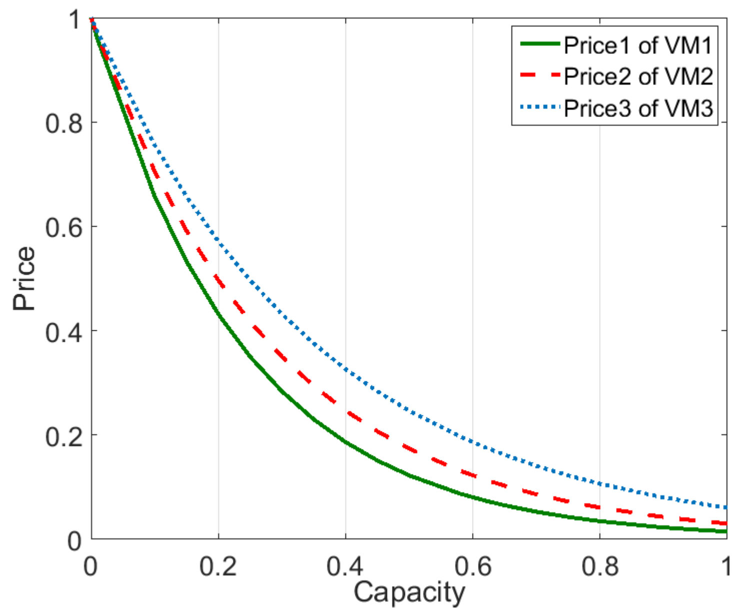


Figure 6.1.1: Prices for multiple classes

price as in Fig. 6.1.1. The prices vary depending on the available capacity at a certain time, as we previously mentioned. Notably, incorporating the operational costs for each VM class is beyond the scope of this thesis. Theorem 3 (Non-Monotone property): $\triangle f(C,t)$ is not necessarily monotone in the available capacity C. Let $\triangle f(C,t)$ be supermodular in C, which represents the remaining capacity of all classes. For simplicity, assume that $C = \{c_1, c_2\}$ is limited to two classes of VMs, (c_1, c_2) and (c_1', c_2') are two states in the cloud, where $c_1 \succeq c_1'$ and $c_2' \succeq c_2$. Since $\triangle f(C,t)$ is supermodular, we can write the following inequality [86].

$$\begin{aligned} \triangle f((c_1, c_2)) + \triangle f((c_1', c_2')) &\leq \triangle f((c_1, c_2) \vee (c_1', c_2')) \\ &+ \triangle f((c_1, c_2) \wedge (c_1', c_2')), \end{aligned} \tag{41}$$

which yields

$$\triangle f((c_1, c_2)) + \triangle f((c_1', c_2')) \leq \triangle f((c_1, c_2')) + \triangle f((c_1', c_2)). \tag{42}$$

Rearranging (42) as follows

$$\triangle f((c_1, c_2)) - \triangle f((c_1', c_2)) \leq \triangle f((c_1, c_2')) - \triangle f((c_1', c_2')) \tag{43}$$

which represents the sufficient condition, increasing differences, for supermodularity. Therefore, $\triangle f(C, t)$ is still monotone similar to its status in one class model. However, suppose $c_1 = c_1' + 1$, $c_2' = c_2 + 1$ and suppose reward of class 1 and 2 are R_1 and R_2 respectively where $R_1 < R_2$ and the probability of selling one VM of class 1 is higher than it is for class 2. Substituting in (43)

$$\triangle f((c_1' + 1, c_2 + 1)) < \triangle f((c_1', c_2 + 1)) \tag{44}$$

and

$$\triangle f((c_1', c_2)) < \triangle f((c_1' + 1, c_2)). \tag{45}$$

Adding (44) and (45) and rearranging gives

$$\triangle f((c_1' + 1, c_2)) - \triangle f((c_1', c_2)) \geq \triangle f((c_1' + 1, c_2')) - \triangle f((c_1', c_2')). \tag{46}$$

This shows decreasing differences, which is a contradiction for supermodularity.

The non-monotonicity property demonstrates the economic phenomenon of rejecting the request when the customer's bid is worthy for a class of VMs and not worthy for another class simultaneously, namely in a single request. The underlying rationale is in the multiple classes system where the request is either entirely accepted or rejected. This behaviour of the total expected revenue implies extra complexity to the problem because the optimal solution may be local not global.

As mentioned earlier in Section 4.2, the approximate optimal solution of (39) can be obtained by accepting any request that holds the following inequality:

$$a_j R_j + \bar{f}(C - A^j, t + 1) \geq \bar{f}(C, t + 1),$$

where $\bar{f} = V^*$. The outline of the proposed cloud revenue maximization using the approximate dynamic programming algorithm is given in Algorithm 2, which is updated version of the Algorithm 1. We refer to this algorithm as Multiple Classes Cloud Revenue Maximization-Dynamic Pricing (MCCRM-DP). MCCRM-DP relies on values of linear approximation of the original dynamic programming values. Note that lines 2 to 4 initialize the maximum number of VMs for each class available for sale, and lines 5 to 7 assign the reward for each class of VM before the announcement of sale. In particular, this reward is evaluated in terms of ρ at the end of horizon, namely the end of sale period $t = T$. Hence the system calculates the expected revenue for each state as stated in line 8. Lines 9 to 13 can be updated every time epoch to satisfy the heterogeneous nature of demands. The one-step probability matrix P_u, also known as the stochastic matrix, represents the probability transitions from any state to others for each decision u. More specifically, it is constructed depending on the expected demand, and the probability to request VM and probability to release it. This can be expressed as a transition from one state to another considering the decision to accept ($u = 1$) and to reject ($u = 0$) the request. Using this general framework, the probability transition matrix in case of accept the request, P_1, can be represented as

$$P_1 = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ \vdots & \vdots & \ddots & \\ p_{k1} & p_{k2} & \cdots & p_{kk} \end{bmatrix}, \tag{47}$$

where p_{nm} is the probability to transfer from state n to state m in case of accepting the request. It is worth mentioning that

$$\sum_{m=1}^{k} p_{nm} = 1 \tag{48}$$

and $p_{nm} \geq 0$.

On the other hand, the probability transition matrix in case of reject the request, P_0, can be represented as an identity matrix $P_0 = \boldsymbol{I}_{kk}$. Line 14 adjusts t to be the period immediately before the end of horizon T. Thereafter lines 16-25 iterate backward to the beginning of horizon $t = 0$. In each iteration, the system computes both the approximated maximum revenue of the current state

Algorithm 2 MCCRM-DP

1: Initialization :
2: **for** each class j, $j = \{1, \ldots, n\}$ **do**
3: $\quad c_j(0) =$ max. number of VMs available for sale
4: **end for**
5: **for** each class j, $j = \{1, \ldots, n\}$ **do**
6: $\quad R_j(T) =$ initial value
7: **end for**
8: Set $f(C, T)$ for all states
9: **for** all $u \in U$ **do**
10: $\quad$ **for** all $c_j \in C$ **do**
11: $\quad\quad$ Set one-step probability transition matrix P_u
12: $\quad$ **end for**
13: **end for**
14: $t = T - 1$
15: Compute:
16: **while** $t \geq 0$ **do**
17: $\quad\quad \bar{f}(C, t+1)$
18: $\quad\quad \bar{f}(C - A^j, t+1)$
19: $\quad$ **if** $(a_j R_j + \bar{f}(C - A^j, t+1) \geq \bar{f}(C, t+1))$ **then**
20: $\quad\quad u = 1$ ”Accept”
21: $\quad$ **else**
22: $\quad\quad u = 0$ ”Reject”
23: $\quad$ **end if**
24: $\quad t = t - 1$
25: **end while**

of cloud resources $\bar{f}(C, t+1)$ and of the next state $\bar{f}(C - A^j, t+1)$ that results from excluding requested VMs from the current state. Line 19 evaluates the worth action, either to sell or to wait for a more promising request in the future. It is worth noting that the arrival of request A^j initiates the system to start the iteration in order to reveal the correct action.

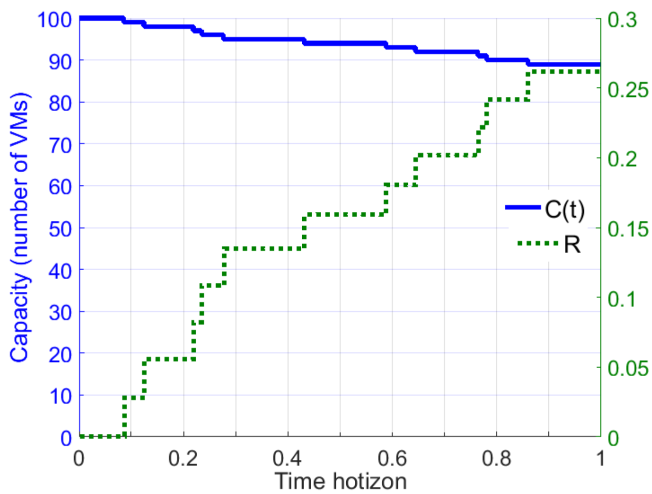
6.2 Performance Evaluation

In this section, we present detailed numerical results of optimal revenue to validate the optimal policy and value function properties. Our discussion addresses two main scenarios, the first restricts cloud’s VMs to one class, which is a coarse scenario, whereas the second considers a cloud of multiple classes of VMs. We provide the expected optimal revenue comparisons through soft and robust demand rates. We also discuss the mutual effect of demand rate on optimal revenue. For both soft and robust rates, we study the dynamics of price over the horizon.

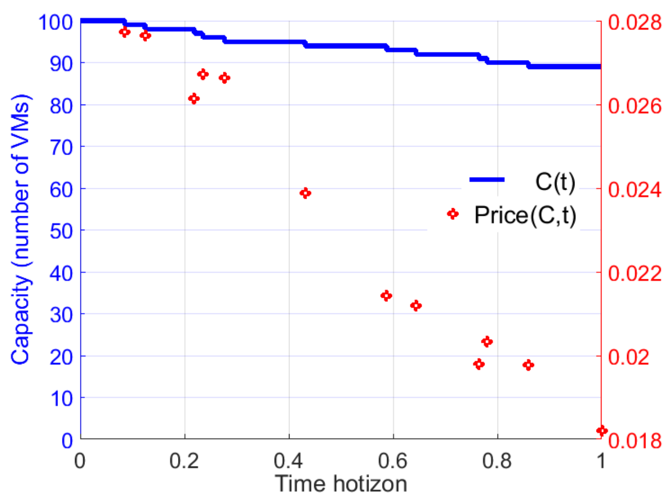
For simplicity, we assume that the system has three classes of VMs $C = \{c_1(t), c_2(t), c_3(t)\}$ with homogeneous demand rates. The maximum quantity allowed to share in spot market is 30, so the cloud could supply up to $C = \{30, 30, 30\}$ for spot market.

6.2.1 Sample Path Revenue

This section proceeds by attempting to answer the question about the values of expected optimal revenue and price in a practical cloud environment. In this regard, we simulate a system of 100 VMs available for spot market with starting normalized prices \$ 0.0277 and \$ 0.030, for low and high rates respectively. They are remarkably low and are nearly equivalent to some classes of Amazon's Spot Instances [4]. The simulation execution, fulfilled many times, generates a random sample path revenue and price for both low and high demand rates using Poisson distributions. Fig. 6.2.1 demonstrates the application of our model in low demand rate $\gamma = 10$. Fig. 6.2.1a shows the expected cumulative revenue within 1 hour horizon in which the customers consume 11 VMs generating the capacity path from 100 to 89 VMs and resulting in \$ 0.261 at the end of horizon. It is clear that the revenue grows as a linear stair-step. Our dynamic pricing scheme records accepted prices as shown in Fig. 6.2.1b. The price level generally decreases over the horizon, from \$ 0.0277 to \$ 0.0182 with local increases in some intervals as in $[0.2, 0.3]$ time interval. The price level increase from \$ 0.0261 to \$ 0.0267 aims to increase the revenue in case of market recovery. Indeed, the price has the same characteristics observed in theoretical analysis, Fig. 4.3.2a.
On the other side, simulating the model considering very high demand rate $\gamma = 120$ achieves cumulative revenue \$ 24.59 at the end of horizon as shown in Fig. 6.2.2a. Further, the revenue seemingly grows exponentially and all stagnant resources, 100 VMs available for spot market, have been utilized before the end of the horizon. Furthermore, the optimal price steadily grows with more differentiation in the second half of the horizon and hence peaks near the end of it as exhibited in Fig. 6.2.2b.



(a) Capacity and reward

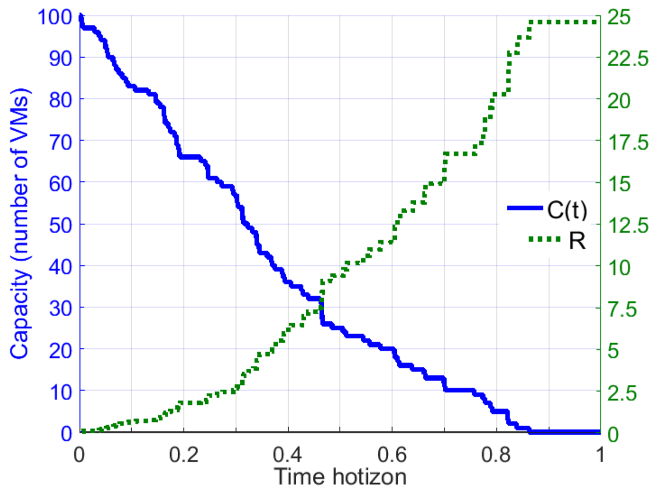


(b) Capacity and price

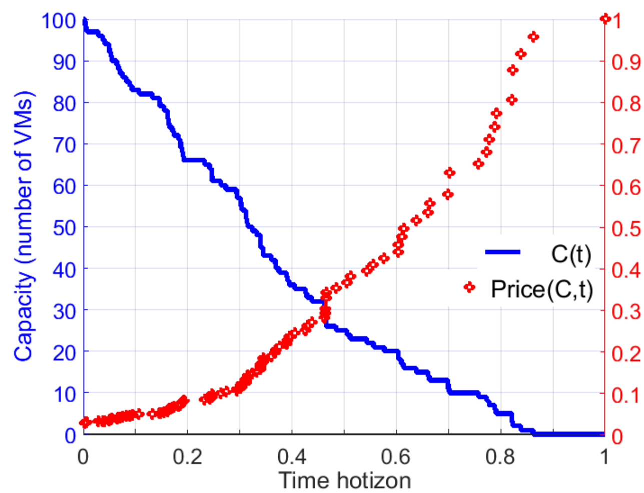
Figure 6.2.1: Random walk, low rate $\gamma = 10$

6.2.2 Impact of Demand Rate

Fig. 6.2.3 present the expected optimal revenue within the last 10 slots for three cases of cloud. It should be easy to note that the demand rates, both low and high, have analogous effects on the optimal revenue as in one class scenario. For instance, Fig. 6.2.3a shows that the expected optimal revenue peaks with approximately $f^*_{\gamma=2}(10, 10, 10, 0.9975) = 5.89$ at time 0.9975, 10 slots before the end of horizon. Then, it rapidly decreases at the end of horizon in case of low demand rate ($\gamma = 2$). Likewise, the expected optimal revenue amounts to $f^*_{\gamma=30}(10, 10, 10, 0.9975) =$



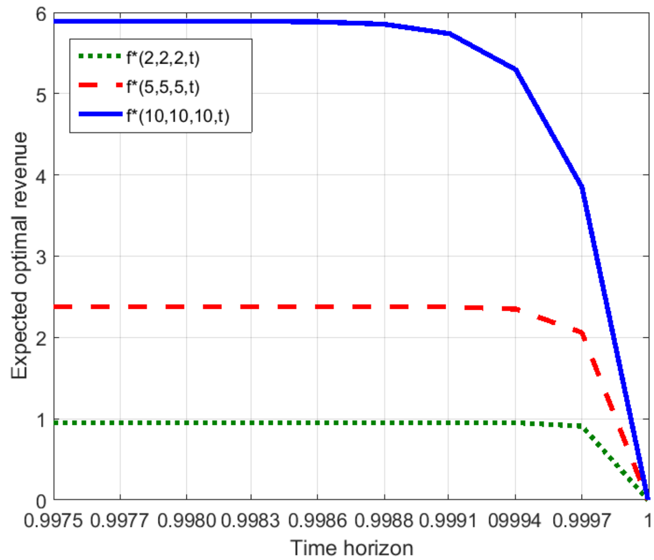
(a) Capacity and reward



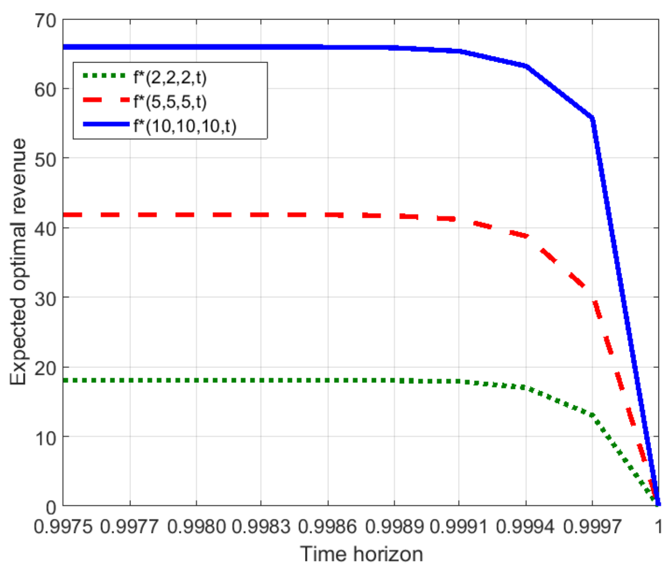
(b) Capacity and price

Figure 6.2.2: Random walk, high rate $\gamma = 120$

65.96 at the same time and decreases at the end of horizon in case of high demand rate ($\gamma = 30$), which is illustrated in Fig. 6.2.3b. At first glance, the variation between these two scenarios seems somewhat insignificant, but closer auditing reveals that multiple classes scenario tends to reap more revenue even at a lower demand ratio. The expected revenue of the total 30 VMs is $f^*_{\gamma=2}(10, 10, 10, 0.9975) = 5.89$ with demand ratio $\frac{2}{30} = 0.06$ in case of multiple classes versus $f^*_{\gamma=10}(50, 0.9975) = 5.55$ with demand ratio $\frac{10}{50} = 0.2$ in case of one class scenario Fig.4.3.1a.



(a) Mean rate $\gamma = 2$



(b) Mean rate $\gamma = 30$

Figure 6.2.3: Expected optimal revenue (Multiple classes scenario)

6.2.3 Expected Optimal Revenue

In this debate, the reader might wonder about the monotonicity of the revenue function. Fig. 6.2.4 depicts the result of expected optimal revenue. When the mean arrival rate $\gamma = 2$, the length of horizon is 3600 intervals, and at the beginning of the horizon the state of the cloud is in the range

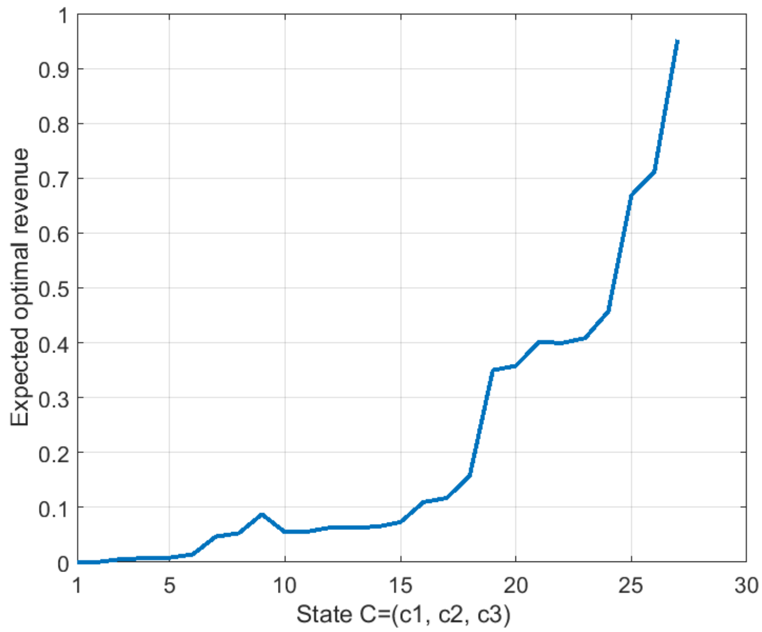


Figure 6.2.4: The optimal revenue with mean arrival rate $(\gamma = 2)$ after $T = 3600$ interval

$C \in [\{0,0,0\},\{2,2,2\}]$. For the sake of clarity, state 1 represents $C = \{0,0,0\}$ and state 26 represents $C = \{2,2,2\}$. The interesting point in this figure, is that the optimal revenue increases in general, but absolutely it is not monotone. Indeed, this legalizes our theoretical and mathematical analysis and proof shown in Section 6.1.3. Moreover, these results coincide with the economic rationale: one class of VMs could raise cloud revenue more than others depending on how much the customer is willing to pay for each class of VMs. To compute the optimal revenue of the current available capacity $C = \{2,2,2\}$, namely the system is in state 26 and Markov decision process implies traversing 26 states. For instance, assume the current state is $C = \{c_1, c_2, \ldots, c_n\}$, then the system traverses

$$S = (\prod_{j=1}^{n}(c_j + 1)) - 1 \tag{49}$$

states. Furthermore, the proposed cloud model has two actions, accept or reject, which implies 2^S possible policies. Finding the best policy among such a massive number of policies is a very significant issue.

6.2.4 Optimal Policy

While managing bids over the course of the horizon, which multiple classes cloud aims to maximize the revenue, it would be too interesting to select the optimal decision u^* (40) that results in the optimal policy π^* as in case of one class scenario. To reveal the optimal policy, we should observe numerical results presented in Fig. 6.2.5. The figure records the results, taking into consideration that the current capacity available for spot market is $C = \{2, 2, 2\}$ and the demand rate $\gamma = 2$. Consequently, the figure shows that the optimal policy π^* is $\{0, 0, 1, 1, 1, 1, 1, 1, 1, 1\}$; since, the demand $(\gamma = 2)$ is low and the supply is high relative to the demand. It is worth noting that in our previous experiments to solve one class of VMs, the optimal policy is to accept in all cases after the first acceptance in any stage if the capacity is above the corresponding first acceptance, Fig. 4.3.4, whereas, in multiple classes of VMs, this scenario does not hold in all cases, such as in stage 3 in Fig. 6.2.5, red marker, and so it does in stage 4. This satisfies the model properties in Section 6.1.3, namely the optimal expected revenue is not monotone.

6.2.5 Comparative Analysis

To expose the extent of the progress have been achieved by our approach (CRM-DP), we study the performance of the most closest approach to ours [97], which we name as Hong. Apart from the multiple classes characteristics, the study includes the evaluation of performance in terms of computing time and price behaviour. For this estimation, we use an Intel Core i7 CPU, 3.6 GHz and 32 GB RAM. First, Fig. 6.2.6a indicates the difference in running time between the two methods. Particularly when the number of stages exceeds 500, Hong's growth is linearly proportional to the stages. Unlike CRM-DP, Hong would then be considered to be impractical in this field because the decision should be fulfilled in real time, as is clear in our method. This confirms our approximation in Section 4.2. Second, in order to evaluate the optimal price behaviour, the experiment assumes that the current pool of stagnant VMs contains 90 VMs out of 100 VMs and the horizon's length is 1 hour like in [97]. To be more credible, the comparison considers merely the behaviour of the optimal price as an evaluating criterion but not the explicit value of the price since the variation in price parameters for each approach. Fig. 6.2.6b shows that Hong's approach almost preserves the

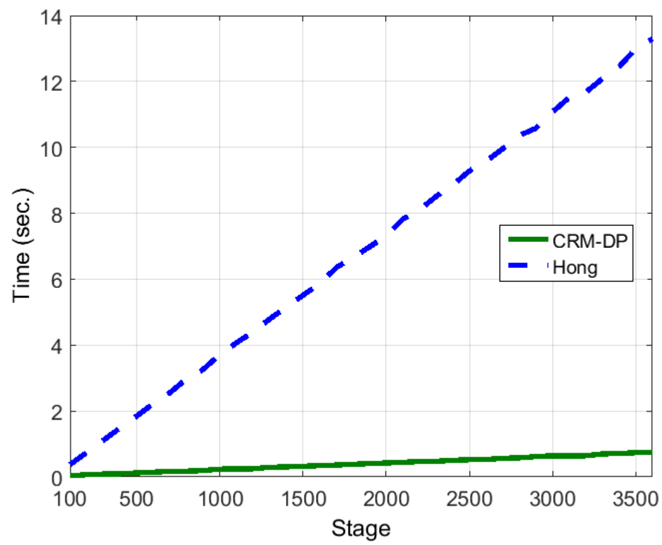
State	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	1	1	1
5	0	0	0	0	0	0	1	1	1	1
6	0	0	0	0	0	1	1	1	1	1
7	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	1	1	1	1	1	1
10	0	0	0	0	1	1	1	1	1	1
11	0	0	0	0	1	1	1	1	1	1
12	0	0	0	0	1	1	1	1	1	1
13	0	0	0	1	1	1	1	1	1	1
14	0	0	0	0	1	1	1	1	1	1
15	0	0	0	0	1	1	1	1	1	1
16	0	0	0	0	1	1	1	1	1	1
17	0	0	0	1	1	1	1	1	1	1
18	0	0	0	1	1	1	1	1	1	1
19	0	0	0	1	1	1	1	1	1	1
20	0	0	0	1	1	1	1	1	1	1
21	0	0	1	1	1	1	1	1	1	1
22	0	0	0	1	1	1	1	1	1	1
23	0	0	1	1	1	1	1	1	1	1
24	0	0	1	1	1	1	1	1	1	1
25	0	0	1	1	1	1	1	1	1	1
26	0	0	1	1	1	1	1	1	1	1
27	0	0	1	1	1	1	1	1	1	1

Stage

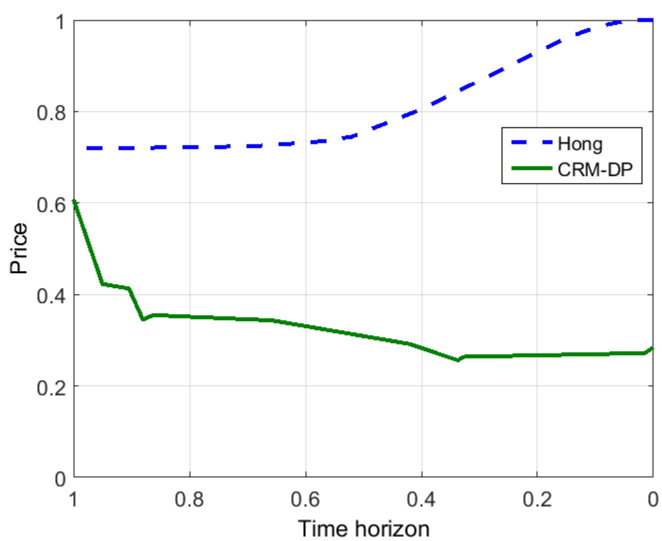
(a) Optimal Policy with $\gamma = 2$

Figure 6.2.5: Optimal policy with entire capacity $C = \{2, 2, 2\}$ and variant rates

opening price through the first half of the horizon then gradually increases the price until it peaks at the end of the horizon. On the other hand, CRM-DP's optimal price decreases and increases occasionally according to the supply and demand principle and time as well. At the beginning, the price decreases in order to optimize sales and then the value is adjusted according to observed demands. However, the price generally decreases over the horizon; since, the demand is very low. Hong's price shows a monotonic increase in both low and high demands, as well as low and high capacities. Moreover, [97] claims that the price is independent of the available capacity in case of high capacity and hence, the time is the dominant factor. In fact, this type of dynamic behavior is more suitable for airline seats than cloud because the seating capacity always decreases over the horizon until the departure time, also referred to as, the end of the horizon, and the time is



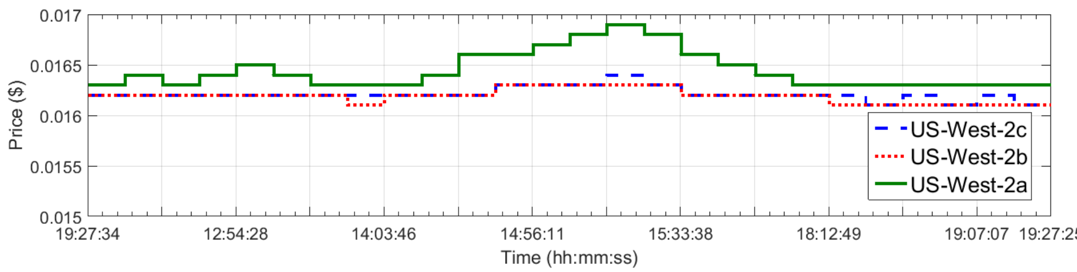
(a) Running time of $f(C = 100)$



(b) CRM-DP price vs Hong price

Figure 6.2.6: CRM-DP vs Hong

the most crucial factor determining the price. However, in cloud the story is different because the capacity, number of stagnant VMs, decreases or increases alike. Subsequently, CRM-DP shows more dynamic and harmonic behavior with the cloud environment because our approach scale up or down the optimal price according to supply and demand principle. As demonstrated in Figs. 6.2.1, and 6.2.2, the price increases when the demand rises or when the supply decreases and it decreases when the demand decreases or when the supply increases.



(a) Amazon VM.

Figure 6.2.7: VM class=c1.medium us-west-2a us-west-2b us west-2c have taken 14 Aug 2016 19:27:34 to 15 Aug 2016 19:27:34

6.3 Discussion

6.3.1 Amazon EC2 Spot Instances

Without a doubt, Amazon Web Services (AWS2) drastically changes IaaS cloud market through it's Spot Instances. It draws cloud providers and research attention to this type of pricing particularly at the outset; since, the information about the adopted mechanism was scarce. While a large body of studies about Spot Instances have focused on discovering Amazon spot pricing strategy, the studies nearly exclusively analysed over a long time of pricing history (year, month, or a week), e.g. [51] and [95]. The overwhelming majority has concluded that the price adjusted artificially or at least it is bounded within predetermined values [51], [31]. To expand on the findings from the previous research, we study pricing history for Amazon specifically after it reveals slightly about its Spot Instances mechanism. The interesting point that attracts our attention is the prices almost vary based on minutes not months, weeks, or even days. Hence, the prices over one day for c1.meduim VM (Linux/UNIX usage) were monitored through three data centres zones us-west-2a, us-west-2b, and us-west-2c. Fig. 6.2.7 shows the prices gathered from 14 Aug 2016 at 19:27:34 to the same time in 15 Aug 2016. It is obvious that,

- The prices are stable for a long time, namely hours, in us-west-2c and us-west-2b, whereas the price varies within minutes in us-west-2a. We deem the demand rates, number of bids, in addition to the capacity of VMs pool, underlie the price changes in a certain time.

- Nevertheless, the price fluctuates between two controlling boundaries.

Further, back to evoke some observations from those whom are interested in this issue [31], [19], unexpected spikes such as the price for m2.2xlarge instances in one zone were found in Sept, 2011. To cope with this debate, Amazon recently discloses some information about its strategy. The customers bid the maximum price they are willing to pay per instance on Amazon's Spot Instances, and the request is accepted if the bid price overrides the presented spot price [3]. Amazon claims that in case all VMs of Spot Instance pool are already utilized, the new customer must bid higher than the current Spot price in order to force the lowest bidder to waive his instance to the new customer. However, from our perspective, this declaration is not convincing. First, according to Amazon's strategy, the customer will bid as close to the current price to save his budget, which contradicts with price spikes; second, it opens the door to manipulations.

6.4 Conclusion

The use of thoughtful spot price of virtual machines has a significant role in contemporary cloud economy because it not only directly related to their revenue, but it also mitigates cloud congestion. Both the uncertainty of arrival and departure demands and the high state complexity pose a challenge to address revenue maximization in real time. Therefore, this chapter proposed MCCRM-DP, a systematic approach towards the challenging aim of cloud revenue maximization, which is specifically tailored towards dynamic price of stagnant VMs of both one class and multiple classes. To inspire, MCCRM-DP we incurred a formulation of profit maximization, using a discrete finite horizon, characterization of optimal properties and controlling conditions, and an approximation of dynamic processes using linear programming. An interesting point of the results is that the multiple class plan is more profitable than one class. Unlike the previous methods of maximizing cloud profitability that focus on feasibility, our MCCRM-DP shows practical and new dynamic analysis insight. Additionally, the numerical results capture and define the optimal policy and demonstrate our analysis. This work amounts to a step ahead in IaaS cloud revenue maximization. It also opens the door to many ideas for future work such as creating more efficient dynamic algorithms to involve different classes of VMs.

Chapter 7

Dynamic resource managment for cloud spot market

After providing dynamic pricing scheme for spot VMs as an efficient pricing strategy for IaaS cloud providers, in this chapter, we generate dynamic resource management system in regards to spot VMs. Lately, there has been a noteworthy interest towards creating resource management systems which ultimately aims to increase the utilization and reduce the waste in the cloud. However, all studies have been achieved in this subject tackle all VMs as being of a homogeneous nature and subject to the same pricing scheme. Our proposed model leverages the distinction between the VMs in terms of its pricing scheme. This chapter exhibits and vividly discusses various aspects of our cloud resource management towards spot pricing scheme model.

Our model is based on spare IaaS cloud resources that we target. Cloud resource management (or resource allocation) considering the entire cloud resources has been extensively researched [37, 66, 84, 96]. In short, spare IaaS resources are non virtualized and unutilized capacities of the power on resources, namely that incur costs during its work. We refer to these resources as stagnant resources (or stagnant capacities) in this section. These resources, are typically distributed over multiple classes of hosts (physical machines) where each host is composed of many components (e.g., CPU, RAM, storage devices (SD), network interface cards (NIC), BIOS, etc.). Stagnant capacities are formed by the imposed conditions, namely service level agreement (SLA) or features provided to the customers. In other words, they can be considered side effects of elasticity, reliability, availability, and security. Studies indicate that these stagnant capacities spend the majority of their life waiting without any revenue and in general most cloud resources experience underutilization dilemma [?], [27], and [84]. Amazon, a pioneer of the spot market in cloud, invents dynamic price for its Spot instances to drive the utilization to a higher levels [?]. Our model strives to locate the optimum resource management for stagnant resources in view of the spot price to maximize cloud revenue.

7.1 Proposed Approach

The block diagram of our proposed dynamic resource allocation approach for spot market is composed of three main components as depicted in Fig. 7.1.1. In a more concise note, the approach

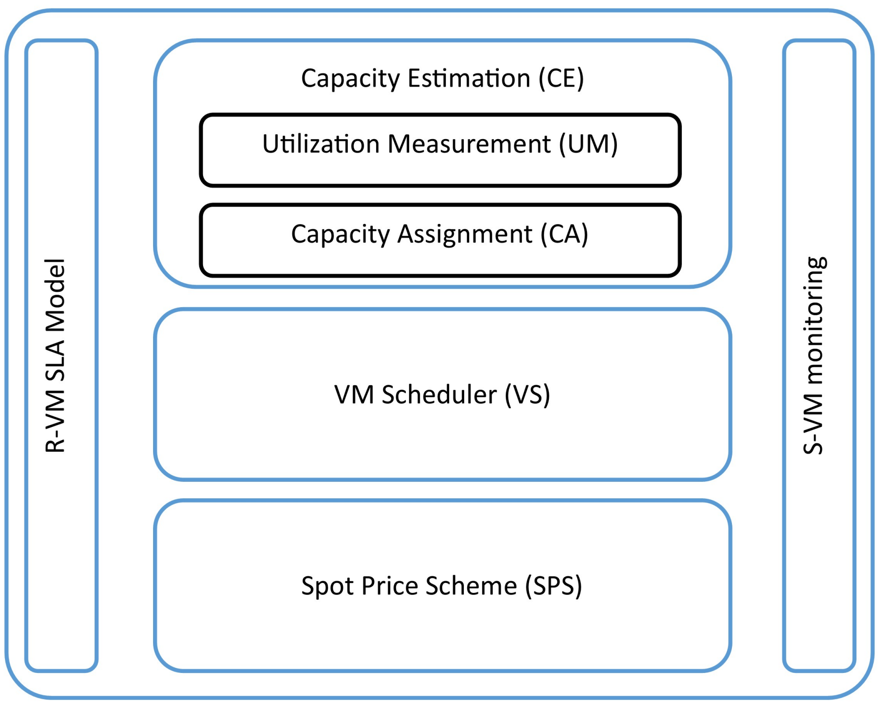


Figure 7.1.1: Block diagram of dynamic resource allocation composed of Capacity Estimation, VM Scheduler, R-VM SLA model, S-VM Monitoring and Spot Price Scheme.

is rendered by estimating accumulated capacity from all hosts using Capacity Estimation (CE) unit.

Subsequently, VM Scheduler (VS) allocates VM combinations that achieve higher marginal revenue. Finally, the Spot Price Scheme (SPS) refers to the optimum decision, migrate, provision, or eliminating the VM, when applicable. The details of the proposed approach are as follow.

7.1.1 Capacity Estimation

In order to estimate the available spare capacity that can be supplied to the spot market, we adopt the technique from [69] that relies on the aggregate capacity. However, [69] apply the aggregate capacity technique for virtual machines that are rely on reserved or on-demand pricing schemes. That is, reserved VMs joint with SLA to supply static price market. We refer to such VMs as R-VMs to distinguish them from spot VMs that are called S-VMs in this paper. These S-VMs are characterized by dynamic price and are not SLA-bound, therefore can be terminated at any time and abandon their sources to R-VMs. In order to achieve the target, we split the task into two phases. In the first, Utilization Measurement scans all hosts to calculate the utilization level for each host separately in addition to the aggregate utilization of all hosts in the cluster. These measurements are used in the second phase, Capacity Assignment, to detect the optimal quantity and allocation of S-VMs throughout the cluster.

The state of the cloud is depicted in Fig. 7.1.2. Obviously, the state of the cloud resources that are currently powered on, shows that reserved virtual machines (R-VMs) consume part of these resources and the rest represent unutilized resources, as denoted by dotted line border. Cloud Estimate (CE) unit estimates and conducts stagnant resources for spot market. It is worth mentioning that managing cloud resources for the reserved market is out of the range of this paper. That is, R-VMs distribution is beyond our system and CE reacts to any change that occurred to the stagnant capacity which results from managing R-VMs or S-VMs alike. In other words, The change of the stagnant capacity could happen either to power on a new VM or to power off VMs for customers from both the reserved and spot markets. The former shrinks stagnant capacity and the latter increases this capacity.

Throughout this section, let κ denote the set of hosts in the cluster, where the number of hosts is $N \triangleq |\kappa|$, and let β denote the set of VMs located in a single host where the number of VMs is

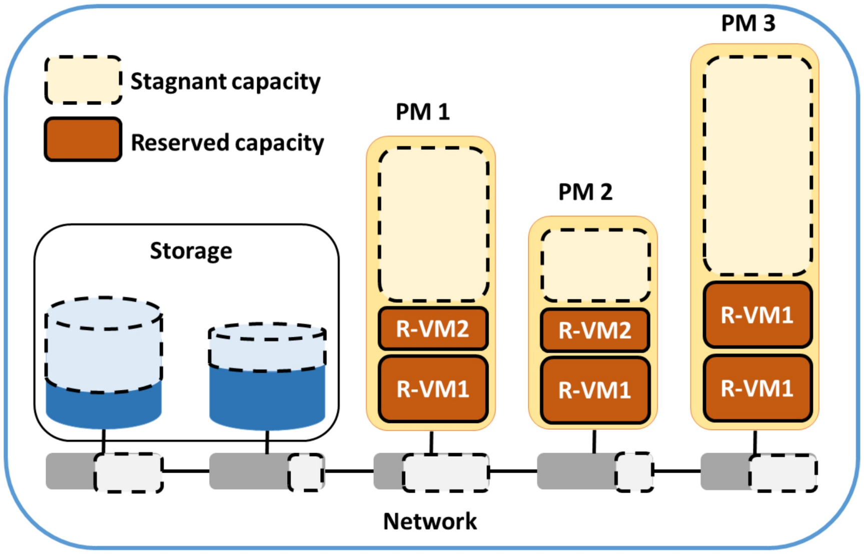


Figure 7.1.2: State of the cloud after deploying R-VMs.

$M \triangleq |\beta|$. For each host $i \in \kappa$, let $c^r_{ij}(t)$ refer to the average utilisation level for virtual machine R-VM$_j$ that is located in the physical machine PM$_i$ at time t. Based on these definitions, the Utilization Measurement (UM) unit calculates the capacity, namely the average utilization level, consumed by all R-VMs that are located in PM$_i$ as

$$C^r_i(t) = \frac{\sum_{j=1}^{M} c^r_{ij}(t)}{M}. \tag{50}$$

Likewise, it calculates the stagnant capacity for each powered on PM$_i$ and the aggregate stagnant capacity of all powered on hosts in the cluster as

$$\begin{aligned} C^s_i(t) &= C_i - C^r_i(t) \\ C^s(t) &= \sum_{i=1}^{N} C^s_i(t). \end{aligned} \tag{51}$$

respectively, where C_i is the entire capacity for PM$_i$. That is, $C^s(t)$ represents the aggregate remaining capacity from all compatible PMs after reserved instances R-VMs consuming $C^r(t)$ from the entire capacity.

The problem concerning packing m S-VMs into stagnant capacity $C^s(t)$ distributed on N host can be formulated as a knapsack problem. The goal is to maximize the revenue f by provisioning the most worthy S-VMs. That is,

$$\begin{aligned}
\max f \quad &= \quad \sum_{i=1}^{N}\sum_{j=1}^{m} p_j y_{ij} \\
&\quad\ \text{s.t.} \quad \sum_{j=1}^{m} c_j^s y_{ij} \leqslant C_i^s \qquad \forall i \\
&\qquad\qquad \sum_{i=1}^{N}\sum_{j=1}^{m} c_j^s y_{ij} > C_i \quad \forall i \\
&\qquad\qquad y_{ij} \in Z_0^+ \qquad (52)
\end{aligned}$$

Where

$$y_{ij} = \begin{cases} number \text{ of S-VM}_j \text{ that are provisioned into host } i \\ 0 \quad otherwise, \end{cases}$$

To achieve the most from $C^s(t)$, each capacity $C_i^s(t)$ must be packed with S-VM$_j$ that has the most profitable price p_j and acceptable capacity c_j^s. We suppose that S-VM$_j$'s capacity c_j^s has two dimensions which represent computing capacity and memory capacity respectively. Further, without loss of generality we assume that S-VMs are sorted so that

$$\frac{p_1}{c_1^s} \geqslant \frac{p_2}{c_2^s} \geqslant \cdots \geqslant \frac{p_m}{c_m^s}.$$

Hence, to solve such an integer linear programming (ILP) problem, in particular that sustain large packages, the Capacity Assignment (CA) unit employs a column generation approach as in [25] and [89]. More specifically a branch and bound technique have been employed. The adoption of column generation approach to solve this problem emphasizes its necessity. Indeed this scheduling problem is of type NP-hard and hitting the global optimum point of an ILP is unsustainable and time-consuming. This technique works by decomposing the problem into two sub-problems, a master LP problem and an auxiliary LP problem (called the pricing problem). Initially, a possible set of

Algorithm 3 Estimating S-VMs

Initialize:
  $C_i^r(t) = 0$ for all PMs
for each PM i, $i = \{1, \dots, N\}$ **do**
  for each R-VM j, $j = \{1, \dots, M\}$ **do**
    $C_i^r(t) = C_i^r(t) + c_{ij}^r(t)$
  end for
end for
for each PM i, $i = \{1, \dots, N\}$ **do**
  $C_i^s(t) = C_i - C_i^r(t)$
end for
S-MAP $\leftarrow$ Using (54), find the optimum number and allocation for S-VMs
return S-MAP

patterns must be identified. The pattern refers to a distinct combination of complementary S-VMs from each class that each PM can host in the cluster. Subsequently, the pricing problem suggests new promising candidates, patterns, to the master problem. If the newly proposed patterns improve the master problem objective, it will be appended to the pattern set as a new column. We refer interested readers to [25] for more details on the aforementioned column generation technique. For instance, Fig. 7.1.3 shows the case of cluster that consists of three PMs and without loss of generality assume each single PM represents merely one host, for the sake of clarity. The UM estimates the stagnant resources, with dotted borders, that cloud be supplied to the spot market, thereafter CA pin packs three classes of VMs in the stagnant capacities so as to aim to maximize the revenue. Table 7.1 shows all possible patterns that could be produced. Note that PM1 leads to three various patterns. However, (51) results only the optimum patterns instead of traversing all possible patterns.
The outcomes of CE are fed to the Spot Price Scheme (SPS) unit to assess the spot price for each S-VM in the next horizon, namely the number of S-VMs from each class that could be involved in the spot market. It is worth mentioning that each cluster consists of compatible PMs regardless of their capability (e.g., Hz of CPU and byte of RAM). The intention is to alleviate the complexity where the proposed approach can work for inhomogeneous clusters separately.

As mentioned previously, this capacity resulted from technical constraints and to meet customers' requirement. One of the most crucial requirements is the SLA between the parties. Therefore selling $C^s(t)$ capacity in spot market and utilizing it through S-VMs must not violate R-VMs' SLA.

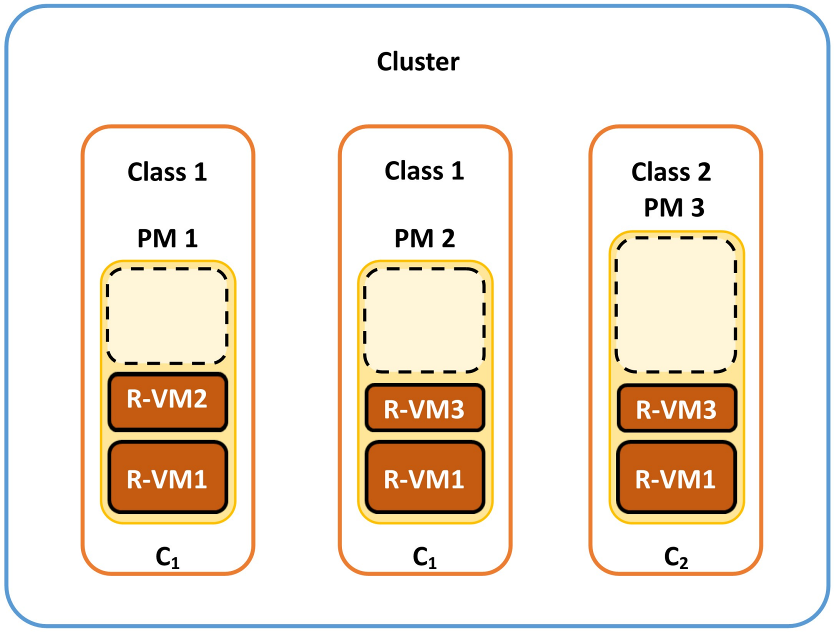


Figure 7.1.3: State of the cluster in the cloud.

All possible patterns					
S-VM	Pt1	Pt2	Pt3	Pt4	Pt5
S-VM1	1	0	0	0	0
S-VM2	0	1	0	0	0
S-VM3	0	1	4	2	1
PM	1	1	1	2	3

Table 7.1: All possible combinations of patterns

7.1.2 R-VM SLA Model

We define an *SLA-violate* function $V(.)$ that is used as basis for determining the possibility of any potential violation for R-VMs as a result of abandoning part of the current capacity in order to reduce the operational expenses. The SLA violation occurs when the following inequality holds for PM$_i$:

$$C^s(t+1) \geq (C^s(t) - C_i)\alpha \tag{53}$$

where $\alpha \in [0, 1]$ is a threshold parameter imposed by the SLA and $C^s(t+1)$ refer to the new stagnant capacity in the next time after dispensing with the capacity of PM$_i$. Therefore,

$$V(x) = \begin{cases} 1 & if\ x\ is\ true \\ 0 & otherwise, \end{cases}, \tag{54}$$

through which the S-VM Scheduler can decide to convert any PM to the sleep mode.

7.1.3 S-VM Scheduler

In order to utilize stagnant resources and get the highest expected return, we split VM Scheduler (VS) into two separate session decisions, where the first decision refers to the computing session and the subsequent decision refers to the networking session. Any noticeable change in stagnant capacity, where noticeable change means equivalent to the minimum size class VM and it is called k in Algorithm 4, initiates VS unit to configure and adjust both PMs and VMs to the optimum status from cloud provider perspective. The computing session elects candidate hosts and S-VMs for termination purposes respecting to SLA, hence the networking session picks out the most convenient host from those candidates according to the networking congestion status. It should be noted here that the working on the networking session is deferred for the future work. Therefore, we assume that the networking resources are available smoothly. The detailed information about VS tasks and its technique is discussed in the following subsections.

Changing the aggregate stagnant capacity $C^s(t)$ refers to the demands and deals that are held under the auspices of the reserved market in time t.

First, releasing or terminating R-VM in reserved market results in increases in $C^s(t)$ amount. This incident induces VS to search for any possible host, PM, without neither R-VM nor S-VM consuming its computing power if any. If such a PM is captured, it can be determined whether it can be discarded from stagnant resources or not in order to reduce cluster's expenses. The rationale of this decision states to convert the proposed PM to the sleep mode for later power off if the abandonment of this redundant capacity does not adversely affect the SLA of R-VMs' customers, otherwise, keep

Algorithm 4 Scheduling S-VMs

```
Given:
C^s(t): the current aggregate stagnant capacity
C^s(t-1): the aggregate stagnant capacity at time t-1
Monitoring: if the change exceeds the threshold k
while (|C^s(t) - C^s(t-1)| >= k) do
    if (C^s(t) - C^s(t-1) > 0) then
        if (is there any PM without any VM) then
            if (terminating the PM violates the SLA) then
                Keep the PM and initiate Algorithm 3
            else
                Convert the PM to the sleep mode
            end if
        else
            if (is there any PM with only S-VMs) then
                if PM's revenue < PM's cost then
                    if (terminating the PM violates the SLA) then
                        Keep the PM and initiate Algorithm 3
                    else
                        Terminate all PMs' S-VMs
                        Convert the PM to the sleep mode
                    end if
                end if
            end if
            Keep the PM and initiate Algorithm 3
        end if
    else
        Initiate Algorithm 5
        Initiate Algorithm 3
    end if
end while
```

it to supply the spot market. Supplying spot market with additional capacity stimulates CE to compute (50) and (51) using Algorithm 3 in order to estimate and pack potential S-VMs. On the other hand, When VS reaches to a PM that hosts only S-VMs, the VS evaluates the revenue from hosting these S-VMs, and then preserves this PM in its current state and stimulates CE to estimate the new possible S-VMs when the revenue exceeds its host's operational costs. However, if the return from these S-VMs is not ample, namely the profit is non positive, the VS checks whether shrinking stagnant capacity by converting the PM to the sleep mode has negative effects on SLA or not. The negative effects on SLA prevents VS from abandoning the PM. In the opposite case, The VS keeps the targeted PM and again initiates Algorithm 3 in order to update S-MAP map.

Algorithm 5 S-VM Monitoring

for each PM i, $i = \{1, \ldots, N\}$ **do**
  $L_i \leftarrow$ workload for each PM$_i$
end for
for each PM i, $i = \{1, \ldots, N\}$ **do**
  while $(L_i > C_i\alpha)$ **do**
    for each S-VM j, $j = \{1, \ldots, \bar{m}\}$ **do**
      TS-VM$\leftarrow$ S-VM with minimum price
    end for
    Terminate TS-VM
  end while
end for

Second, decreasing $C^s(t)$ refers to when a new R-VM has been provisioned in the reserved market or to host, PM, failure causing its resources are no longer available. This incident drives VS to notify S-VM Monitoring unit to terminate S-VMs that could be rival R-VMs on PM's resources. The selection process for terminating S-VMs is based on the price, where the least price S-VM is terminated. To achieve this, the S-VM Monitoring unit applies Algorithm 5. Then, the VS notifies CE unit to reconfigure the stagnant capacity and relocate the pool of S-VMs over PMs to achieve the optimal placement.

7.1.4 S-VM Monitoring

The S-VM Monitoring unit is responsible for monitoring the workload and utilization for each host in the cloud. If the workload of any host exceeds the predefined threshold, this unit terminates the S-VM that generates the lowest revenue first to mitigate the workload and so on until the workload of targeted PM becomes in the allowable range. The rationale behind this is to provide the PM's resources to more worthy S-VMs, namely those with higher revenue, or to R-VMs in case of physical machine failures or for maintenance purposes.

7.1.5 Spot price scheme

The main purpose of this unit is to assign the initial dynamic price for S-VM in the spot market. First, the CE provides the estimated volume for each capacity c_j^s and hence the Spot Price Scheme (SPS) use it to set the initial price p_j. The initial price is determined by the average of many values obtained from SPS for the same capacity state, where initial price p_j is inversely proportional to the stagnant capacity. It worth noting that there is no known distribution of VMs [51] prices specifically in the spot market. This is because the spot price is actually the bidding price which is influenced by the information available to both the consumer and the provider. Nevertheless, the SPS scales the price up and down based on the law of supply and demand. To accomplish this, we employ the dynamic pricing scheme from [23] to control the price over the horizon, namely the time look ahead is one hour. The dynamic pricing scheme is formulated as a finite horizon stochastic Markov decision process.

On the other hand, the VS induces the SPS to provide the updated spot prices in order to terminate undesirable S-VMs and free up spaces for the most profitable S-VMs.

7.2 Experiments

In this section, we present the empirical evaluation findings. We closely follow the steps and techniques of the proposed approach, dynamic resource allocation for spot market (DRASM), that were discussed in Section 7.1 to show the results and effects of dynamic allocation in the view of aggregate capacity.

7.2.1 Technical specification

In this assessment, we implement Cloudsim [34] as a simulation tool and modelling framework for our provisioning technique. The cloud data center is characterized by 12 hosts (or PMs) and a dynamic workload normally distributed with mean value equal to the normal workload in real data centers. Without loss of generality, we assume that the cloud contains two types of hosts and the specifications for the hosts are given in Table 7.2. The PMs consolidate 12 virtual machines of 4 classes with different configurations as shown in Table 7.3. Moreover, we consider spot prices of T2 Amazon Spot instances price and all bidding prices for S-VMs are equal or more than the setting-up prices.

Host	Features
HP ProLiant ML110 G4	Xeon 3040 1860 MHz, 2 cores, 4 GB
HP ProLiant ML110 G5	Xeon 3075 2660 MHz, 2 cores, 4 GB

Table 7.2: Hosts specifications

Virtual machine	MIPS	RAM
S-VM1	2500	1870
S-VM2	2000	1740
S-VM3	1000	1740
S-VM4	500	613

Table 7.3: VMs specifications

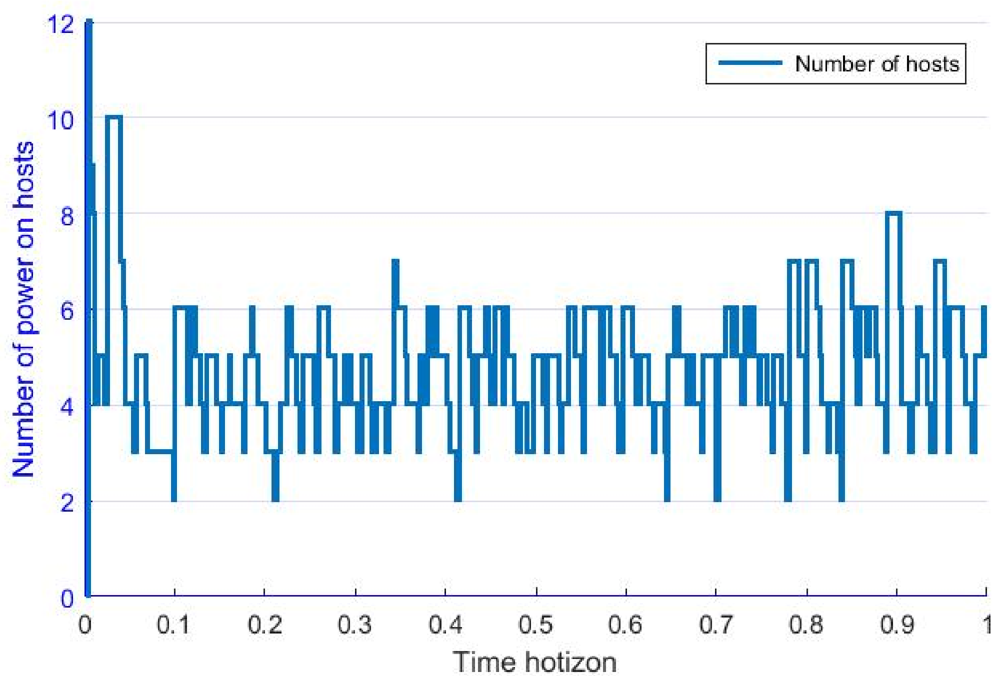


Figure 7.2.1: Active hosts

7.2.2 Numerical results

Figure 7.2.1 depicts the findings of the average of 10 executions for our model using Cloudsim. The results show that, the average number of powered on hosts within one hour interval is approximately 5 hosts, while at the beginning of the period it is 12 hosts. It is clear that the capacity saving is 58%. The interesting point about the results is that the significant saving refers not only to

S-VMs's pool but also to the activities of R-VMs, where the R-VMs consume merely about 25% of cloud capacity. [69] demonstrates that the average capacity saving is around 40% considering only R-VM virtual machines. This is attributed to the fact that S-VMs are volatile (when the host contains highly loaded R-VMs) virtual machines compared to R-VMs. Next, we look at the utilization level of powered on hosts.

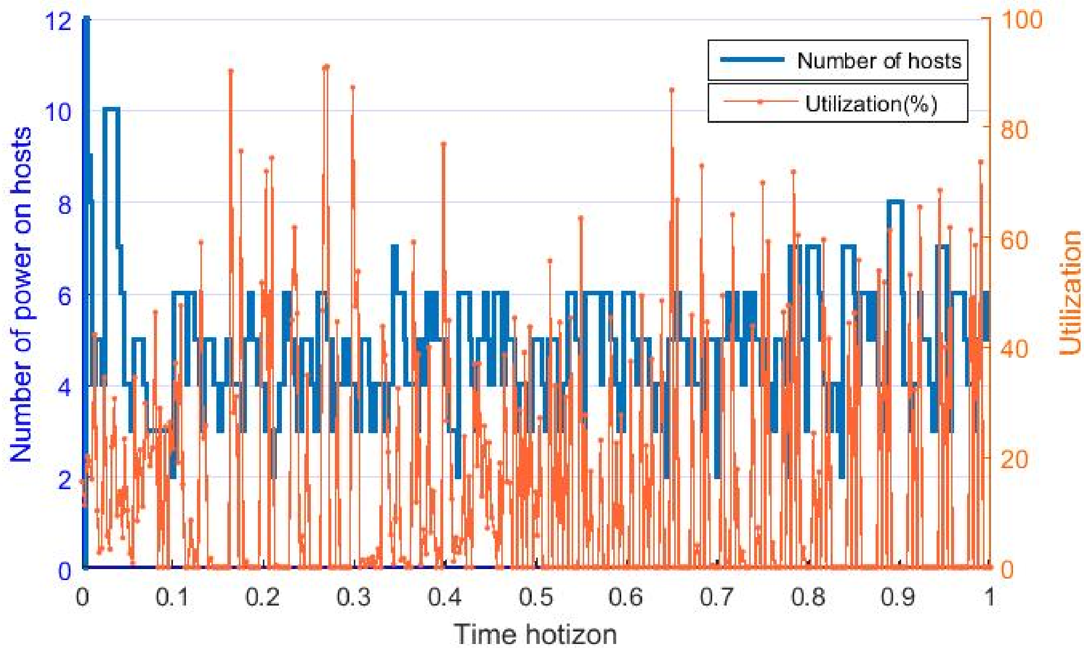


Figure 7.2.2: Active hosts and utilization

The results of monitoring the utilization level of hosts are shown in Figure 7.2.2. The figure shows the average utilization for each host (i.e., each time slot represent the utilization of 12 hosts from host-1 until host-12 respectively) within one hour. A high utilization level is observed at $t = 0.16$, $t = 0.28$, and $t = 0.66$, which indicates that some hosts are overloaded. The rationale of this is that some S-VMs attempt to consume a much greater quantity of its aggregate capacity whereas the individual capacity for these S-VMs enabling them to consume capacities more than its aggregated capacity. It is also clear that in the subsequent time slots the utilization levels become under 80% which validate our algorithm approach, because the cloud must terminate the least profitable S-VM to release more capacities for either R-VMs or S-VMs virtual machines. Further, we notice that the utilization level peaks when the number of hosts becomes low (e.g., here it is between 4 and 6

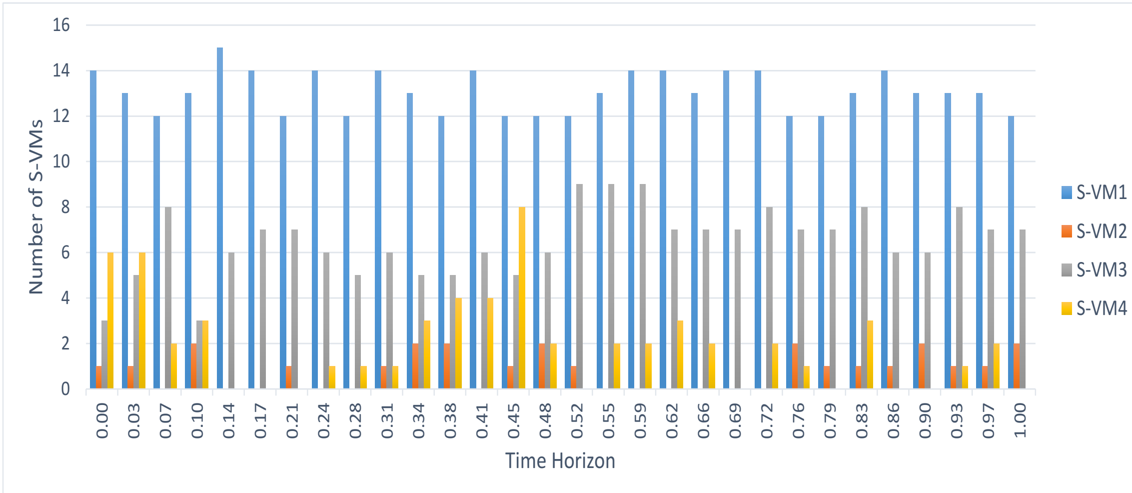


Figure 7.2.3: S-VMs hosting configurations

hosts). This case refers that the cloud provider reduces supplied resources (or capacities) to the spot market.

Figure 7.2.3 illustrates the distribution of S-VMs classes over cloud hosts during the analyzed period (1 hour). In an effort to maximize cloud marginal revenue through spot market, we compute the available capacity for each host, then we find the best hosting configuration for all S-VM classes depending on the price. The outcome shows that at time $t = 0.10$ the best hosting configuration is 13 of S-VM1, 2 of S-VM2, 3 of S-VM3, and 3 of S-VM4 while at time $t = 0.17$ the best hosting configuration is 14 of S-VM1 and 7 of S-VM3.

7.2.3 Performance evaluation

Please note that we do not claim that aggregate capacity technique and dynamic resource allocation are contributions of this section, nevertheless, we have to pinpoint that to the best of our knowledge this is a first of dynamic resource allocation in cloud that considers separate categories of virtual machines, namely S-VMs and R-VMs. Therefore, it is not possible to compare with what has been achieved in this area are very cumbersome. Consequently, we evaluate our proposed approach, dynamic resource allocation for spot market (DRASM), merely from a utilization perspective of all VM categories. To this end we present a comparison between our proposed approach DRASM and MPC, presented by [100]. To fulfil this target we use the workload dataset from Google's Compute

Clusters [48], which represent the resource consumptions for both CPU and memory for 7 hours. We simulate IaaS cloud infrastructure similar to that used to implement MPC which consists of 7000 PMs where all PMs contain identical CPUs of 10640 MHz. Further, we adopt three classes of VMs which match the specifications of [100] as shown in Table 7.4. Figure 7.2.4 depicts the

Virtual machine	MIPS	Average duration (sec.)
S-VM1	2000	14049
S-VM2	1000	4862
S-VM3	500	1694

Table 7.4: VMs experimental setup

first hour of utilization for all VM classes considering that 173751 distinct requests for VMs are serviced. It is clear that at time $t = 0.5$ the average utilization level is 53% and 39% for MPC and DRASM respectively. The rationale behind this variance is that MPC allocates fixed capacity for each class of VMs during the horizon while DRASM allocates aggregated capacity for each class of VMs. Another interesting point about these findings is that in DRASM 3000 PMs are converted to the sleep mode during the experiment hence the value of utilization, 39%, refers to 4000 hosts instead of 7000 hosts as in MPC's experience. However after time $t = 0.7$, the utilization for both approaches reach very close levels. This status rises when each VM is highly overloaded. In other

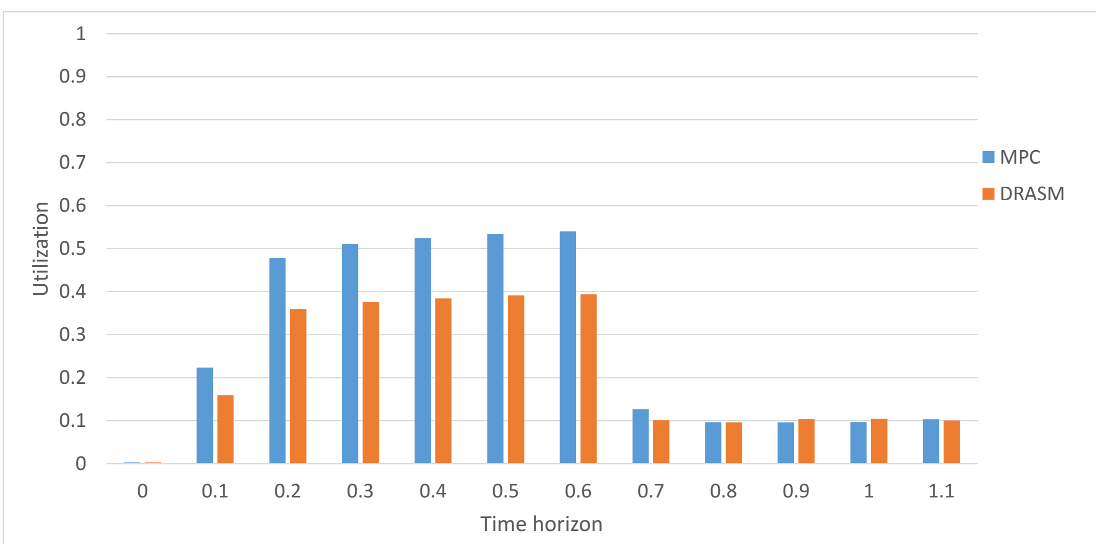


Figure 7.2.4: DRASM vs MPC

words, the aggregate capacity of VMs equates the cumulative maximum capacity for VMs.

7.3 Conclusion

This chapter presents a new approach of dynamic resource allocation for spot market (DRASM). It aims at minimizing hosting costs as well as maximizing cloud revenue through creating an efficient resource management technique to consolidate spot market's VMs. First, the capacity estimation (CE) unit determines the candidate capacity for sale in the spot market. Then, it creates the optimal mapping between S-VMs and PMs in view of the revenue. This mapping results from applying column generation approach. Second, the VM scheduler (VS) unit monitors and controls S-VMs pool so as to increase cloud's hosts utilization and to decrease hosting costs on the other hand. The section leverages and applies several techniques to fulfil the desired tasks, aggregate capacity, dynamic resource allocation, and dynamic pricing. Empirical evaluations accomplished using real requests with their workload showed that the findings exhibit promising hosting savings.

Chapter 8

Conclusions and Future Work

The ever increasing communication speed and adoption of Internet applications has been great values both socially and economically. The full embracing of Internet technologies allowed the innovation and implementation of creative thoughts that facilitate everyday tasks. Indeed cloud is one of those creative ideas which has attracted the attention of users and investors alike. The main challenge for cloud providers is to convince users to adopt the cloud as a medium for their IT activities. However, recent studies and techniques demonstrated that cloud could be managed, configured and deployed in methods that lead to a great usefulness for both parties.

This thesis was dedicated to address the ever increasing unutilized capacities concern rendered by demand, service level agreement, and elasticity, which are the core source of wasting high amount of spare capacity in IaaS cloud infrastructure. We tackled the concern by employing dynamic pricing strategy and dynamic resource management that consequently maximize IaaS cloud revenue.

In this thesis, firstly, we reviewed the literature in terms of cloud pricing tiers, pricing techniques and strategies, and more particularly spot price methods in IaaS cloud. Consequently, we focused our attention to research work towards the problem of creating a dynamic pricing scheme for spare resources. To this end, we proposed a formulation of revenue maximization using a discrete finite horizon, characterization of optimal properties and controlling conditions, and an approximation of dynamic processes using linear programming. The formulation considered stochastic arrival and departure demands and one class of VMs. Our proposed model and CRM-DP algorithm show practical and new dynamic analysis insights. After creating such dynamic pricing scheme, we attempted to mitigate the high dimensionality of the model using column generation technique. Further, we characterized dynamic pricing necessities, objectives, and parameters for inference purposes. Our endeavour to develop a system that meets the practical needs of cloud providers led us to develop our previous dynamic pricing scheme to cover multiple class of VMs. One interesting point of our findings, related to the dynamic pricing scheme of multiple class VMs, is that the multiple class plan is more profitable than one class. The last problem that we tackled in this thesis, namely dynamic resource allocation, was rendered by inferring the necessity for creating a particular resource management system for spot VMs, which is a promising target for IaaS cloud providers.

It is important to refer to the limitations related to the proposed methods. In fact, These restrictions define the area in which the proposed system is effective and on the other hand be a starting point for future study:

- Number of VMs: In a One Class System, the number of VMs can range from 100 to 1000, while in a Multiple classes system, the number of VMs can only be a few hundreds. However, the proposed approaches are aimed at selling the VMs that represent the spare capacities in the spot market, Which are of course few compared to the size of the total cloud.
- The estimation process of both the arrival and departure rates associated with a given price plays an important role in the sensitivity of the experimental results. This also means the requirement for an effective estimation approach to get the optimal results.

From the completed research, we can extract the following issues and research challenges for future research:

- Development and innovation of advanced spot pricing metrics can be considered as an extension for this thesis. For instance, inferring and attributing the price by solely monitoring the stagnant capacity of cloud resources requires more scrutiny and verification from real-cloud markets. This work can reveal more pricing metrics that should be considered.

- Study and analyze customers reaction and how they switch between various pricing schemes in IaaS cloud computing environments. Cloud revenue management systems must investigate the diversion of customers to other pricing schemes.

- Our proposed resource management approach for spot market did not address the availability of network resources and the best alternative technique.

- Resource allocation management approach dedicated for spot market is still not thoroughly investigated in the literature and hence necessitates more attention from the researchers to fill this gap and find effective models for this target.

- Despite the existence of many dynamic pricing strategies projects, more price and revenue resources and information sharing efforts should arise in order to contribute to the development of studies and academic research in this regard.

As for future work, we aim at address the following research and development points:

- The design and achievement of resource allocation management approaches for spot market that consider live migration for spot VMs os as to maximize cloud revenue.

- Extend the work of this thesis to include handling the competitors' prices.

Bibliography

[1] *2015 Review Shows $110 Billion Cloud Market Growing at 28% Annually*. https://www.srgresearch.com/articles/2015-review-shows-110-billion-cloud-market-growing-/28-annually. [Online; accessed April-2016].

[2] *Amazon EC2 Pricing*. http://aws.amazon.com/ec2/pricing/. [Online; accessed February-2015].

[3] *Amazon EC2 Spot Instances*. https://aws.amazon.com/ec2/spot/. [Online; accessed June-2016].

[4] *Amazon EC2 Spot Instances Pricing*. https://aws.amazon.com/ec2/spot/pricing/. [Online; accessed May-2017].

[5] *Amazon just revealed it has a profitable, $6 billion beast of a business*. http://www.businessinsider.com/aws-earns-1-billion-a-year-on-6-billion-in-profit-2015-4. [Online; accessed May-2017].

[6] *Azure pricing*. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/. [Online; accessed February-2015].

[7] *Cloud Computing Server Utilization the Environment*. https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/. [Online; accessed April-2016].

[8] *Cloud VPS servers*. http://www.vps.net/. [Online; accessed March-2015].

[9] *Gartner Says Worldwide Public Cloud Services Market to Grow 18 Percent in 2017*. http://www.gartner.com/newsroom/id/3616417. [Online; accessed May-2017].

[10] *How Spot Fleet Works*. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet.html. [Online; accessed June-2016].

[11] *How Spot Instances Work*. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/how-spot-instances-work.html. [Online; accessed July-2016].

[12] *Preemptible Virtual Machines*. https://cloud.google.com/preemptible-vms/. [Online; accessed May-2016].

[13] *Pricing Philosophy*. https://cloud.google.com/pricing/philosophy/. [Online; accessed May-2016].

[14] *Server Decommissioning Workshops*. https://uptimeinstitute.com/uptime_assets318e311aec592d3b1f38cbf0ff96ffd09b-00110A.pdf. [Online; accessed January-2017].

[15] *Spot Instance Pricing History*. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html. [Online; accessed April-2016].

[16] *Virtual server.net*. https://www.virtual-server.net/home/. [Online; accessed March-2015].

[17] *Virtual Servers*. http://www.softlayer.com/virtual-servers. [Online; accessed March-2015].

[18] *We power the Microsoft Cloud*. http://www.microsoft.com/en-us/server-cloud/cloud-os/global-datacenters.aspx. [Online; accessed June-2016].

[19] *What to do when Amazon's spot prices spike*. https://gigaom.com/2011/12/27/how-to-deal-with-amazons-spot-server-price-spikes/. [Online; accessed July-2016].

[20] *What's New*. http://aws.amazon.com/about-aws/whats-new/2009/. [Online; accessed March-2015].

[21] Mukul Agarwal, Vivek S Borkar, and Abhay Karandikar. Structural properties of optimal transmission policies over a randomly varying channel. *Automatic Control, IEEE Transactions on*, 53(6):1476–1491, 2008.

[22] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Schuster Assaf, and Dan Tsafrir. Deconstructing Amazon EC2 Spot instance pricing. *ACM Transactions on Economics and Computation*, 1(3):1–20, 2013.

[23] Fadi Alzhouri and Anjali Agarwal. Dynamic pricing scheme: Towards cloud revenue maximization. *IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 168–173, 2015.

[24] Fadi Alzhouri, Anjali Agarwal, and Yan Liu. Maximizing cloud revenue using dynamic pricing of multiple class virtual machines. *IEEE Transactions on Cloud Computing*, pages 1–1, 2018.

[25] Fadi Alzhouri, Anjali Agarwal, Yan Liu, and Ahmed Saleh Bataineh. Dynamic pricing for maximizing cloud revenue: A column generation approach. In *Proceedings of the 18th ACM International Conference on Distributed Computing and Networking (ICDCN)*, pages 22:1–22:9, 2017.

[26] Sebastian Anthony. *Microsoft now has one million servers – less than Google, but more than Amazon, says Ballmer*. http://www.extremetech.com/extreme/161772-microsoft, 2013. [Online; accessed August-2016].

[27] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, 2009.

[28] R. Bellman and Rand Corporation. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

[29] Richard Bellman, Robert Kalaba, and Bella Kotkin. Polynomial approximation–a new computational technique in dynamic programming: Allocation processes. *Mathematics of Computation*, 17(82):155–161, 1963.

[30] A. Beloglazov and R. Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24:1366–1379, 2013.

[31] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 spot instance pricing. In *IEEE Third International Conference on Cloud Computing Technology and Science*, pages 304–311, 2011.

[32] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[33] Gérard P Cachon and Pnina Feldman. Dynamic versus static pricing in the presence of strategic consumers. *Working paper, University of Pennsylvania*, 15:2011, 2010.

[34] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41:23–50, 2011.

[35] Q. Chen, P. Grosso, K. van der Veldt, C. de Laat, R. Hofman, and H. Bal. Profiling energy consumption of VMs for green cloud computing. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 768–775, 2011.

[36] Shigang Chen and K. Nahrstedt. Hierarchical scheduling for multiple classes of applications in connection-oriented integrated-service networks. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, pages 153–158, 1999.

[37] Papagianni Chrysa, Leivadeas Aris, Papavassiliou Symeon, Maglaris Vasilis, Cervello-Pastor Cristina, and Monje Alvaro. On the optimal allocation of virtual resources in cloud computing networks. *IEEE Transactions on Computers*, 62(6):1060–1071, 2013.

[38] Vasek Chvatal. *Linear Programming*. Series of books in the mathematical sciences. William H. Freeman, 1983.

[39] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[40] Linna Du. Pricing and resource allocation in a cloud computing market. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 817–822, 2012.

[41] Kenneth J. Duda and David R. Cheriton. Borrowed-virtual-time (bvt) scheduling: Supporting latency-sensitive threads in a general-purpose scheduler. *SIGOPS Oper. Syst. Rev.*, 33:261–276, 1999.

[42] Jeffrey Galloway, Karl Smith, and Susan Vrbsky. Power aware load balancing for cloud computing. *Proceedings of the World Congress on Engineering and Computer Science*, I:19–21, 2011.

[43] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[44] A. Gohad, N. C. Narendra, and P. Ramachandran. Cloud pricing models: A survey and position paper. In *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–8, 2013.

[45] Geoffrey J Gordon. Approximate solutions to Markov decision processes. *Ph.D. thesis, Carnegie Mellon University*, page 228, 1999.

[46] G. Gratzer. Lattice Theory: First Concepts and Distributive Lattices. Dover Publications, 2009.

[47] Reem Heakal. *Economics Basics: Supply and Demand*. http://www.investopedia.com/university/economics/economics3.asp. [Online; accessed April-2015].

[48] Joseph L. Hellerstein. Google cluster data, January 2010. Posted at http://googleresearch.blogspot.com/2010/01/google-cluster-data.html.

[49] Jianhui Huang and Dan MA. *The pricing model of Cloud computing services*. http://ink.library.smu.edu.sg/sis_research/1742. [Online; accessed April-2016].

[50] Kevin Jackson. *OpenStack Cloud Computing Cookbook*. Packt Publishing, 2012.

[51] Bahman Javadi, Ruppa K Thulasiram, and Rajkumar Buyya. Statistical modeling of Spot instance prices in public cloud environments. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 219–228. IEEE, 2011.

[52] Bahman Javadi, Ruppa K. Thulasiram, and Rajkumar Buyya. Characterizing spot price dynamics in public cloud environments. *Future Generation Computer Systems*, 29(4):988 – 999, 2013.

[53] Hai Jin, Xinhou Wang, Song Wu, Sheng Di, and Xuanhua Shi. Towards optimized fine-grained pricing of IaaS cloud platform. *IEEE Transactions on Cloud Computing*, 3(4):436–448, 2015.

[54] Nan Jin, Gayathri Venkitachalam, and Scott Jordan. Dynamic congestion-based pricing of bandwidth and buffer. *IEEE/ACM Transactions on Networking (TON)*, 13(6):1233–1246, 2005.

[55] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, pages 39–50. ACM, 2010.

[56] Julian Keilson and Adri Kester. Monotone matrices and monotone Markov processes. *Stochastic Processes and their Applications*, 5(3):231–241, 1977.

[57] Andreas Krause and Daniel Golovin. *Submodular Function Maximization*. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014.

[58] Bhavani Krishnan, Hrishikesh Amur, Ada Gavrilovska, and Karsten Schwan. VM power metering: Feasibility and challenges. *SIGMETRICS Perform. Eval. Rev.*, 38(3):56–60, 2011.

[59] Andrei Krokhin and Benoit Larose. Maximizing supermodular functions on product lattices, with application to maximum constraint satisfaction. *SIAM Journal on Discrete Mathematics*, 22(1):312–328, 2008.

[60] Sanjay Kumar, Vanish Talwar, Vibhore Kumar, Parthasarathy Ranganathan, and Karsten Schwan. vManage: Loosely coupled platform and virtualization management in data centers. In *Proceedings of the 6th International Conference on Autonomic Computing*, pages 127–136. ACM, 2009.

[61] U. Lampe, M. Siebenhaar, A. Papageorgiou, D. Schuller, and R. Steinmetz. Maximizing cloud provider profit from equilibrium price auctions. In *IEEE Fifth International Conference on Cloud Computing*, pages 83–90, 2012.

[62] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

[63] Z. Lee, Y. Wang, and W. Zhou. A dynamic priority scheduling algorithm on service request scheduling in cloud computing. In *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, pages 4665–4669, 2011.

[64] Richard Lipsey and Colin Harbury. *First principles of economics*. Oxford University Press, 1992.

[65] Mario Macías and Jordi Guitart. A genetic model for pricing in cloud computing markets. In *Proceedings of the ACM Symposium on Applied Computing*, pages 113–118, 2011.

[66] Syed Hamid Hussain Madni, Muhammad Shafie Abd Latiff, Yahaya Coulibaly, and Shafi'i Muhammad Abdulhamid. Resource scheduling for infrastructure as a service (iaas) in cloud computing. *J. Netw. Comput. Appl.*, 68(C):173–200, 2016.

[67] Sameer Kumar Mandal and Pabitra Mohan Khilar. Efficient virtual machine placement for on-demand access to infrastructure resources in cloud computing. *International Journal of Computer Applications*, 68, 2013.

[68] Ishai Menache, Ohad Shamir, and Navendu Jain. On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 177–187. USENIX Association, 2014.

[69] Xiaoqiao Meng, Canturk Isci, Jeffrey Kephart, Li Zhang, Eric Bouillet, and Dimitrios Pendarakis. Efficient resource provisioning in compute clouds via vm multiplexing. In *Proceedings of the 7th International Conference on Autonomic Computing*, pages 11–20, 2010.

[70] Antony Messerli. *Method and System for Utilizing Spare Cloud Resources*. http://www.google.com/patents/US20130247034, 2013. [Online; accessed April-2016].

[71] R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45:65–72, 2012.

[72] Ioannis A. Moschakis and Helen D. Karatza. Evaluation of gang scheduling performance and cost in a cloud computing system. *The Journal of Supercomputing*, 59:975–992, 2012.

[73] Pradeep Padala, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem. Adaptive control of virtualized resources in utility computing environments. *SIGOPS Oper. Syst. Rev.*, 41:289–302, 2007.

[74] Ioannis Ch Paschalidis and John N Tsitsiklis. Congestion-dependent pricing of network services. *Networking, IEEE/ACM Transactions on*, 8(2):171–184, 2000.

[75] G. Baley Price. Definitions and proporties of monotone functions. *Bulletin of the American Mathematical Society*, 46:77–80, 1940.

[76] I. Ruiz-Agundez, Y. K. Penya, and P. G. Bringas. A flexible accounting model for cloud computing. In *2011 Annual SRII Global Conference*, pages 277–284, 2011.

[77] J. Savill. *Mastering Hyper-V 2012 R2 with System Center and Windows Azure*. Wiley, 2014.

[78] Soumya Sen, Carlee Joe-Wong, Sangtae Ha, and Mung Chiang. Incentivizing time-shifting of data: a survey of time-dependent pricing for internet access. *Communications Magazine, IEEE*, 50(11):91–99, 2012.

[79] Moshe Shaked and J. George Shanthikumar. Stochastic orders and their applications. Academic Press, 1994.

[80] Barry Smith, John Leimkuhler, and Ross Darrow. *Yield management at American airlines*. *Interfaces*, 22:8–31, 1992.

[81] S. Son and K. M. Sim. A price- and-time-slot-negotiation mechanism for cloud service reservations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):713–728, 2012.

[82] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 10–10. USENIX Association, 2008.

[83] F. Teng and F. Magoules. *Resource Pricing and Equilibrium Allocation Policy in Cloud Computing*. In *10th IEEE International Conference on Computer and Information Technology*, pages 195–202, 2010.

[84] Luis Tomás and Johan Tordsson. Improving cloud infrastructure utilization through overbooking. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, pages 5:1–5:10, 2013.

[85] A. N. Toosi, K. Vanmechelen, K. Ramamohanarao, and R. Buyya. Revenue maximization with optimal capacity control in infrastructure as a service cloud markets. *IEEE Transactions on Cloud Computing*, 3(3):261–274, 2015.

[86] Donald M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.

[87] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis. Flexible use of cloud resources through profit maximization and price discrimination. In *2011 IEEE 27th International Conference on Data Engineering*, pages 75–86, 2011.

[88] Konstantinos Tsakalozos, Herald Kllapi, Eva Sitaridi, Mema Roussopoulos, Dimitris Paparas, and Alex Delis. Flexible use of cloud resources through profit maximization and price discrimination. In *IEEE 27th International Conference on Data Engineering (ICDE)*, pages 75–86, 2011.

[89] S. Vakilinia, B. Heidarpour, and M. Cheriet. Energy efficient resource allocation in cloud computing environments. *IEEE Access*, 4:8544–8557, 2016.

[90] Cheng Wang, Neda Nasiriani, George Kesidis, Bhuvan Urgaonkar, Qian Wang, Lydia Y. Chen, Aayush Gupta, and Robert Birke. Recouping energy costs from cloud tenants: Tenant demand response aware pricing design. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*, pages 141–150. ACM, 2015.

[91] Wei Wang, Ben Liang, and Baochun Li. Revenue maximization with dynamic auctions in IaaS cloud markets. In *IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*, pages 1–6, 2013.

[92] Elizabeth Louise Williamson. *Airline network seat inventory control: Methodologies and revenue impacts*. Technical report, Cambridge, Mass.: Massachusetts Institute of Technology, Dept. of Aeronautics & Astronautics, Flight Transportation Laboratory, 1992.

[93] Cynara C. Wu and Dimitri P. Bertsekas. Admission control for wireless networks. `http://www.mit.edu/~dimitrib/Adcontrol.pdf`. [Online; accessed August-2016].

[94] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24:1107–1117, 2013.

[95] Hong Xu and Baochun Li. Maximizing revenue with dynamic cloud pricing: The infinite horizon case. In *IEEE International Conference on Communications (ICC)*, pages 2929–2933, 2012.

[96] Hong Xu and Baochun Li. Anchor: A versatile and efficient framework for resource management in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, 24:1066–1076, 2013.

[97] Hong Xu and Baochun Li. Dynamic cloud pricing for revenue maximization. *Cloud Computing, IEEE Transactions on*, 1(2):158–171, 2013.

[98] C. S. Yeo and R. Buyya. Managing risk of inaccurate runtime estimates for deadline constrained job admission control in clusters. In *2006 International Conference on Parallel Processing (ICPP'06)*, pages 451–458, 2006.

[99] Sangho Yi, Derrick Kondo, and Artur Andrzejak. Reducing costs of Spot instances via checkpointing in the Amazon elastic compute cloud. In *Cloud Computing (CLOUD), IEEE 3rd International Conference on*, pages 236–243, 2010.

[100] Q. Zhang, Q. Zhu, and R. Boutaba. Dynamic resource allocation for spot markets in cloud computing environments. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pages 178–185, 2011.

[101] Lidong Zhou Zhengping Qian. *Distributed Systems Meet Economics: Pricing in the Cloud.* In *HotCloud '10, USENIX Workshop on Hot Topics in Cloud Computing*, 2010.