2017

Efficient Pricing Methods in The Cloud Computing Market

Soleimani, Maryam

http://hdl.handle.net/11023/3629
master thesis

University of Calgary graduate students retain copyright ownership and moral rights for their thesis. You may use this material in any way that is permitted by the Copyright Act or through licensing that has been assigned to the document. For uses that are not allowable under copyright legislation or licensing, you are required to seek permission.

Downloaded from PRISM: https://prism.ucalgary.ca
UNIVERSITY OF CALGARY

Efficient Pricing Methods in The Cloud Computing Market

by

Maryam Soleimani

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA
January, 2017

© Maryam Soleimani 2017
Abstract

The emerging cloud computing paradigm enables cloud systems to provide multiple heterogeneous types of cloud resources for end customers over a network. Users and providers in these systems attempt to maximize their revenue using well designed pricing methods. Auctions are considered as efficient mechanisms for resource sharing and charging users in cloud systems.

We study the online social welfare maximization problem at a cloud market, and design efficient pricing functions to be used in online auction mechanisms for cloud resource provisioning, for tasks with completion deadlines. Combining the techniques of primal-dual approximation algorithm design with our proposed pricing methods, we design a cloud auction that runs efficiently in polynomial time, guarantees truthfulness, and achieves near-optimal social welfare, in the cloud eco-system. Simulation studies confirm the efficacy of the proposed mechanism.
Acknowledgements

I would like to gratefully acknowledge various people who have helped me during the completion of my thesis. This thesis would not have been completed without the help and support from all those people.

First and foremost, I would like to sincerely thank my supervisor, Dr. Zongpeng Li. I cannot express my appreciation and thanks enough, that this thesis would not have been possible without his help. Thanks for giving me the opportunity to study at the University of Calgary, and providing me guidance and encouragement when there were difficulties in my research. I am so grateful for his innovative ideas and consistent precious feedback throughout this work. He has been a role model to me with his passion and talents in research. In addition, Dr. Li has not been only my teacher, but also been my mentor in other aspects of my life, helped me to adapt to the new environment in Calgary when I first came here, and helping me pointing out the direction for my future career. I am sincerely thankful for the opportunity to work with him.

To my parents, they have been supporting me and caring about me all the time. During the time I felt frustrated and lonely, they comfort and encourage me and when I had doubt about myself and my progress they always give me the motivation to continue my work and do my best. Without them, I would not have the determination and courage to pursue master degree abroad.

To my lab mates, who have offered me much help over these two years. Thanks for discussing with me when I had questions in my course work or research.

Finally, many thanks to the department of Computer Science for all the financial support they have provided and the administrative staff in the department for their guidance.
List of Tables

2.1 Rules for constructing the dual program [1], . . . . . . . . . . . . . . . . . . . 18
2.2 Possible combinations for the primal and its dual [1], . . . . . . . . . . . . . 20
2.3 Different pricing models for cloud computing infrastructure [2], . . . . . . . 24
4.1 Summary of notation in cloud auction system model . . . . . . . . . . . . . 36
4.1 $f(x) = x \cdot e^x + b$ for $c = 2$ and $b = 2.5$ .............................................. 45

5.1 Competitive ratio of fixed pricing function .......................................................... 56
5.2 Competitive ratio of linear pricing function ......................................................... 57
5.3 Competitive ratio of quadratic pricing function for different $U_r/L_r$ ............... 58
5.4 Competitive ratio of logarithmic pricing function for different $U_r/L_r$ .......... 60
5.5 Competitive ratio of pricing function presented by Zhou et al. for different $U_r/L_r$ .. 61
5.6 Competitive ratio of exponential pricing function for different $U_r/L_r$ ........... 63
5.7 Competitive ratio of all proposed pricing functions for $U_r/L_r = 30$ ............... 64
5.8 Competitive ratio of all proposed pricing functions for $U_r/L_r = 30$ ............... 64
5.9 User satisfaction of quadratic pricing function for different $U_r/L_r$ ................. 67
5.10 User satisfaction of logarithmic pricing function for different $U_r/L_r$ .......... 67
5.11 User satisfaction of pricing function presented in [3] for different $U_r/L_r$ .... 68
5.12 User satisfaction of exponential pricing function for different $U_r/L_r$ .......... 68
5.13 Social welfare of the Alg 1. with different number of users and different pricing functions ................................................................. 69
5.14 Social welfare of quadratic pricing function for different $U_r/L_r$ ................. 71
5.15 Social welfare of logarithmic pricing function for different $U_r/L_r$ ............... 71
5.16 Social welfare of exponential pricing function for different $U_r/L_r$ ............... 72
## List of Symbols, Abbreviations and Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA</td>
<td>Resource Allocation</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Program</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Program</td>
</tr>
<tr>
<td>OPT</td>
<td>Optimal Solution</td>
</tr>
<tr>
<td>VCG</td>
<td>VickreyClarkeGroves</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter introduces the context of the research area explored in this thesis. We start with an overview of the environment of the problem i.e., cloud computing systems, and continue with the motivation behind this research work, including resource allocation and payment mechanisms in the cloud market. We further provide discussions on the contributions and outline of the thesis.

1.1 Overview

Cloud computing has recently emerged as a new paradigm offering on-demand network access to computing resources such as CPU, RAM and storage. This technology offers scalable on-demand sharing of resources and costs among a large number of cloud users. In addition, this system enables users to manage and process data efficiently with reasonable prices. Many definitions have been provided for cloud computing in [4] [5] [6]. According to Foster et al. [6], cloud computing is defined as “a large-scale distributed paradigm that is driven by economies of scale, where a pool of virtualized, dynamically-scalable, managed computing storage, power and services are delivered on demand to external users over the internet.”

In today’s Internet, there is an increasing demand for the services that cloud systems provide using multiple datacenters in different geographical locations. These services can be provisioned rapidly and released with minimal management overhead [7]. Services offered to the users in cloud computing include [8]: infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS), storage as a service (STaas), test environment as a service (TEaas), security as a service (SECaaS) and many more. In general, resources are located in a datacenter that is shared among multiple clients, dynamically assigned
and shared according to users’ demands. In today’s cloud computing systems, resource allocation plays a critical role in the overall performance of the entire system and the level of customer satisfaction. At the same time, the service provider needs to secure its own profit [9]. Thus, the resource allocation mechanism can be seen as a means to ensure that the users’ requirements will be processed correctly while considering the current status of each resource in the datacenter [10].

The main goal of a cloud provider is to maximize its revenue using its employed pricing methods. Simultaneously, users’ goals are to achieve the highest level of utility with a reasonable price. Thus, an optimal pricing methodology is required to satisfy both sides’ goals. The pricing function for charging cloud users is one of the important metrics controlled by the cloud provider in order to encourage usage of cloud services [8]. The pricing method is a critical factor for these systems, since the pricing function can affect customer behaviour, loyalty to the provider and overall system revenue and utility, including provider’s revenue and users’ utilities. Developing appropriate pricing methodology together with an effective allocation rule, results in an efficient mechanism for resource leasing in cloud computing system.

1.2 Motivation

Auction mechanisms provide an efficient approach for resource allocation in the cloud market, considering the finite resource supply in providers and the fact that careful allocation of resources is important to achieve high social welfare. Auction mechanisms contain two main components: Allocation rule and Payment rule. In the allocation rule, efficient resource allocation algorithms are used to allocate resources among cloud users. Based on the payment rule, the users will be charged accordingly for the received services. Currently, most of the cloud providers adopt a fixed price policy in order to determine user payments. Unlike the simple allocation methods based on fixed pricing, auction mechanisms are able
to automatically discover the real-time market prices based on demand-supply, and provide resources to those users who value them the most, resulting in an economically efficient approach \[11\]. Besides, the cloud resource allocation needs to be managed at the time when users’ demands arrive. For the sake of serving multiple users in a dynamic environment, an extension of classic mechanism design is used, namely online auction mechanisms. While the arrival of users has an arbitrary manner, the auctioneer makes an irrevocable decision for each bid upon its arrival, whether to serve the user or not, without any knowledge of future requests. Thus, modeling the procedure using an online method will meet the need of making decisions on the requests by the time users arrive.

Most of the existing work on online auction mechanisms only considers flat value based methods, in which users present uniform valuations on resource units during their presence in the online auction mechanism and will be charged by fixed pricing methods. In many time-critical applications, however, the value of the resource depends on time of delivery, and in case of late delivery the value of the requested resource would decrease \[12\].

In this thesis, we focus on designing multiple marginal pricing functions for an online allocation mechanism for cloud computing resource sharing based on resource demands with completion deadlines. The cloud provider acts as the auctioneer who owns the cloud resources in the datacenter and leases the resources to the users through the auction mechanism. In this model, each user has a start time which is her arrival time to the cloud system, as well as a departure time that is the resource release time upon job completion. After resource allocation is completed, the auctioneer needs to determine the payment of each user. This payment can be determined by our presented pricing functions.

1.3 Contributions

In this thesis, we focus on an online auction mechanism to study the cloud market for computing jobs, each of which has a completion deadline, with multiple resources requested
by users. In this model, the cloud provider acts as the auctioneer and owns the computing resources such as CPU, RAM and storage. The provider offers the services to the users, who acts as bidders, through an auction. In the online setting, bids arrive sequentially, each to be accepted or rejected immediately. The cloud user bids for future cloud resources to execute her tasks. Each bid includes a deadline specifying the preferred completion time of the requested task and a valuation, reflecting the amount that the user is willing to pay for execution of the job.

As the first step in the cloud market design, we formulate the social welfare maximization problem as an integer linear program, which is proven to be NP-hard. Besides, there is a challenge caused by deadline constraints. To address the challenge of deadline constraints, we borrow the new technique of compact exponential-size LPs with dual separation oracles, presented by Zhou et al. in [3]. While the formulated social welfare maximization problem is still NP-hard, we resort to the primal-dual design technique to address this challenge. Recall that the auction mechanism consists of two main components: allocation rule and payment rule. For the allocation rule, we adapt the resource scheduling algorithm proposed by Zhou et al. [3] and modify it to be applicable to our problem formulation. This algorithm is based on the primal-dual framework, which enables us to compute near-optimal solutions for the problem. Then, for the payment rule, we propose three different marginal pricing functions, which are compatible with the online nature of the auction mechanism, and charge users according to the problem criteria. Another challenge in designing auction mechanisms is guaranteeing truthfulness. Truthfulness is a desirable property in auction mechanism design. To guarantee truthful bidding from cloud users, it is necessary to design an accompanying payment methodology for eliciting truthful bids that work in concert with resource allocation algorithms. The presented pricing functions in our work calculate a distinct payment for each possible time slot in which a user may win a resource. Later, it is shown that these pricing functions can guarantee truthfulness.
Then, we evaluate the online algorithm using each pricing function, and show that it executes in polynomial time, which results in computational efficiency. Moreover, each of the three proposed pricing functions computes the users’ payments regardless of users’ bidding price, resulting in the truthfulness property, which is proved later in this thesis. We examine the competitive ratio of the algorithm using the proposed price functions and show that the algorithm achieves a near-optimal competitive ratio for one of the pricing functions. Therefore, the pricing function in cooperation with the online allocation algorithm will make an efficient online auction mechanism.

Our contributions in this thesis can be summarized as follows:

- We provide a system model for resource allocation in the cloud market, which covers the system constraints and users’ demands. The model considers multiple resources shared among users with requests for tasks with completion deadlines.

- We design marginal pricing functions for charging users in this cloud market, which results in an online efficient auction mechanism when working in concert with an adopted allocation algorithm.

- We conduct theoretical analysis of the proposed pricing functions to show that they guarantee truthfulness. Moreover, the competitive ratio analysis is provided for one of the pricing functions, which is proven to have a small competitive ratio.

- We conduct simulations to study the performance of the auction mechanism with proposed pricing functions. The online mechanism is evaluated by simulation studies for its social welfare, user satisfaction and competitive ratio.

- Further, we measure and compare the competitive ratio of all our proposed pricing functions with an existing payment function proposed by Zhou et al. [3]
and the fixed pricing method. The evaluation results, similar to theoretical analysis, show a better competitive ratio achieved by one of our pricing functions.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 of this thesis is devoted to reviewing the background and important research findings relevant to the subject of this research. First, theoretical fundamentals on auction theory and approximation algorithm design are discussed in order to build a consistent terminology to be used throughout this thesis, and review the theoretical principles and foundations. Next, resource allocation and pricing models in cloud computing are covered. At the end of this chapter, the process of auction method simulation studies and data generation techniques are provided. In Chapter 3, previous literature related to this thesis is reviewed.

Chapter 4 is dedicated to presenting the cloud auction mechanism, including the cloud auction system model and bidding language and the primal-dual algorithm for resource sharing, which is modified for our system model. Then, our proposed pricing functions are presented in order to determine the payments by users. We then provide the theoretical analysis for our proposed pricing functions, including the analysis of the truthfulness property, and for one of the presented pricing functions, a competitive ratio is proven.

In Chapter 5, the simulation studies and performance evaluation of the method are provided. The experiments are conducted using synthesized data according to the system properties and assumptions. The performance evaluation results are presented for different criteria of the system including social welfare, competitive ratio and user satisfaction. Then, the corresponding results are demonstrated and discussed. Chapter 6 summarizes and concludes the thesis. We also propose research directions to expand as future work. Appendix A provides the mathematical steps for our system model and the simulation codes for algorithm
implementation is provided in Appendix B.
Chapter 2

Background

In this chapter, we review the principles and foundation of auction theory and mechanism design, in order to build a consistent terminology to be used in the rest of the thesis. Moreover, resource allocation methods and pricing models in cloud based infrastructures will be discussed. Section 2.1 focuses on the principles of auction theory, and multiple auction types are discussed later in this section. Section 2.2 includes linear programming formulation and the procedure of Dual Program construction. Section 2.3 focuses on resource allocation in cloud computing and describes strategies and challenges for resource allocation in these systems. Pricing models in the literature of cloud computing are covered in Section 2.4. At the end of this chapter, possible simulation methods and data generation techniques for auction methods are covered.

2.1 Auction Theory

2.1.1 Auctions

Auction theory has been recognized as one of the most widely practiced fields in economic and game theory in recent years. This theory concerns the design of auctions and their set of governing rules. Note that a subtle change in auction rules can result in significant differences in the outcome of the auction. Thus, auction theory deals with studying the efficiency of auction designs for different sets of rules.

An auction is a kind of economic activity that represents the behavior of buyers and sellers in the activity. Formally, the participants of an auction include the bidders who are willing to buy an item in a competition with other bidders, and the auctioneer who runs the auction and provides specific rules for allocation and payment.
Auctions are considered as mathematical games from the game theoretic perspective, in which: 1) the bidders and auctioneers are considered as players; 2) the bidders’ strategies are to maximize their utilities; 3) payoff of each bidder is defined as her expected utility; 4) the auctioneer sets the rules according to her objective value, which is in the form of cost minimization or profit maximization; and 5) each bidder has a set of actions, which is her bid function [13] [14].

Different auctions are able to promote different kinds of behavior among bidders. We consider the situation in which a seller is trying to sell a single item to multiple bidders, and runs an auction among a set of bidders each of whom is interested in buying the item [14]. The competition among bidders is through presenting bids. Bids express the bidders’ willingness for paying monetary amounts for different items, which is private information for each bidder. In auctions, bidders are assumed to be rational, i.e., they wish to choose the possible strategy that results in the best possible outcome. Such a game is known as a non-cooperative game in which rational players are capable of making decisions independently [13].

A non-cooperative game reaches the state of Nash Equilibrium, when under the assumption that other players do not change their bids, none of the players is better off by changing her strategy unilaterally. If a strategy gives as good or better result as any other strategy, regardless of other opponents strategies, it is called a dominant strategy. The strategy is weakly dominant, if there is at least one set of opponents’ actions where this strategy has a superior outcome. A strategy is called a strictly dominant strategy if it always provides a better result than any other strategy, regardless of what other opponents offer. Therefore, if there is a strictly dominant strategy available for a player, the player will apply that strategy in each Nash equilibrium of the game. Note that weakly dominated strategies are also capable of constituting a Nash equilibrium.
2.1.2 Mechanism Design

Mechanism design is a sub-field of economic theory with an engineering perspective. The goals of the designed mechanisms are referred to as social choice. A social choice is an aggregation of the preferences of different players toward a single joint decision. Mechanism design is considered as an economic mechanism that influences users’ behavior by giving them incentives for choosing desirable actions, which is one of the main techniques for controlling multi-agent systems [15].

An auction mechanism is defined by a set of rules including: 1) an auction protocol that includes the syntax, sequences and semantics of messages and social choices; 2) payment rules for calculating the payment for the winners of our auction mechanism; 3) allocation rules which consider the allocation constraints and constraints related to the overall objective function [16]. The goal of the auction is related to the auction’s outcome, and can be revenue maximization for the seller, or social welfare maximization that maximizes the total utility of all players, including the seller and the buyers.

2.1.3 Valuation of Bidders

Each bidder has a valuation of the requested items/resources, which is represented in a valuation function; this function is a real-valued function, representing each user’s private valuation of each item of type $r$, shown by $v_{ir}$ or the valuation of $i$th-user of the whole amount of requested resources, shown by $v_i$. It is assumed that the bidders’ valuation functions have two following properties:

1) The function is normalized, meaning that $v(0) = 0$.
2) The function is monotone.

According to the valuation function, the utility of users can be defined as $u_i = v_i - p_i$, where $p_i$ is the price that $i$th-bidder should pay. Note that the bidders’ valuations are private to each bidder and the value does not depend on the valuation of other bidders. This
mechanism is called *independent values*.

Vickrey [17] showed that there is a pricing method for a private valuation model when there is a single item or multiple items to be sold, in which a winner can never affect the charged prices. Therefore, the bidders have no incentive to misreport their valuations. Thus, we can have a truthful auction by making it a dominant strategy for users to report their values truthfully.

**Bidding Languages**

Bidding language is a set of interpretation rules and communicated message formats, which are used by bidders to formulate their bids. Logical bidding languages allow a bidder to create complex bids in which the logical structure of the utility function is captured. Examples of bidding languages are as follows [18] [19]:

1) **Atomic or Single-Minded Bids**: This bidding language has the format of \((r, p(r))\) where \(r\) is the item that the bidder is willing to bid on, and \(p(r)\) is the bid valuation of the user [18].

2) **Exclusive OR (XOR Bids)**: Exclusive OR bid is the submission of an arbitrary number of atomic bids. Each bid \((r_i, p(r_i))\) represents \(i\)th-user’s bid, in which \(r_i\) is a subset of items requested by the user, and \(p(r_i)\) is the maximum price bidder \(i\) is willing to pay for the requested item. The XOR of two bids is to take the best one of them, but not both. In this case, the bidders are willing to achieve at most one of the bids they submit [18] [19].

3) **OR Bids**: This type of bidding is also a collection of an arbitrary number of atomic bid pairs \((r_i, p(r_i))\), \(r_i\) is the subset of items and \(p(r_i)\) is the maximum price user \(i\) is willing to pay for the subset of items. The bidder is willing to achieve any number of atomic bids for the sum of their reported prices [19].

4) **OR-of-XORs and XOR-of-ORs**: OR of XORs represents OR of a set of XOR bids, while XOR of ORs represents XOR of a set of OR bids. Note that OR-of-XORs and XOR-of-ORs languages are incomparable in their expressive power. In fact, there are bids that can be expressed concisely in the OR-of-XORs language but require exponential size XOR-of-ORs
bids, and conversely, there are bids that can be expressed concisely in the XOR-of-ORs language but need exponential size OR-of-XORs bids [19].

2.1.4 Auction Types

Auctions can have different types depending on the rules governing their mechanism design. Auctions can be categorized as two types: written (sealed bid) auctions, and oral (open) auctions. Sealed-bid auctions for single items were developed within traditional auction theory. An underlying assumption in modeling these auctions is that each bidder has an intrinsic value for the item, which is referred to as the bidder’s true value for the item being auctioned. Specifically, she is willing to buy the item for a price up to this value. On the other hand, oral auctions are those where bidders are present, hearing other bidders’ bids and making decisions [14].

There are four main forms of auction types:

1) **Ascending-bid Auction (English Auction) [14]**: This type is one of the oldest and most frequent auction forms, which is considered as an oral auction. In this type of auction, the seller increases the price of the item gradually. Bidders drop out until only one of them remains, which will be considered as the winner of the auction who will pay the final price at the auction.

2) **Descending-bid Auction (Dutch Auction) [20]**: Unlike the English auction, in this type the seller decreases the price gradually from a high initial value until a bidder accepts the price and buys the item with the current price.

3) **First-price sealed-bid Auction [21]**: In this auction, bidders submit their sealed bids to the auctioneer without knowledge of any of their opponents’ bids. After receiving the bids, the auctioneer unseals the bids and selects a winner. The bidder who submits the highest bid is awarded the item and pays her bid.

4) **Second-price sealed-bid Auction [21]**: This auction is known as Vickrey auction, in which the bidder who submits the highest bid is awarded the item, and pays the amount
of the second highest bid.

Auctions can be categorized based on features other than the rules that they are governed by, such as their environment, including the number of sellers and buyers or the number of goods being sold. This category is discussed in the following:

1) **Single Item Auctions** [22]: In this type of auction, there is a single item/good available for selling to the buyers. If multiple units of the same item/good are available to be traded, the auction is called *multi-unit single item auction*. Note that multi-unit auctions provide facilities for negotiations on large quantities of the same item.

2) **Multi-item Auctions** [23]: If the auctioneer provides several items/goods for the auction procedure, then the auction is called a multi-item auction. In this type of auction, it is possible to achieve the minimum equilibrium price allocation by performing a dynamic auction rather than a single round auction [13]. Note that sealed-bid multi-item auctions are considered as a generalization of Vickrey auctions, if each bidder is interested in obtaining at most one item. In this case each user submits a sealed bid including her valuation of all the requested items. Its drawback is that it fails to identify that a bidder’s valuation for a combination of items is more or less than the sum of individual items’ value.

3) **Combinatorial Auctions** [24]: In this type of auction, bidders are able to place bids on combinations of items/goods, called *packages*, rather than individual items. This approach enables users to express valuations on the packages of items, which results in more precise preferences and leads to greater auction revenue and improved economic efficiency. Moreover, it is an important feature when items are complements. Items are complements when a set of items has greater utility than the sum of the utilities for each individual item, which is considered as a drawback of multi-item auctions.

Combinatorial auctions were first proposed by Rassenti *et al.* [25] for the allocation of airport landing slots. These auctions are motivated by the question that “what if there is a set R of $m > 1$ items to be auctioned off to $n$ players, with each desiring a subset of them?”
An idea is to run a separate second-price type of auction for each item. This approach works if each user $i$ has a separate value for each item, while the value of subset $r \subseteq R$ of items to user $i$ is the summation of its value for the items of $r$. One drawback of using this approach is that it ignores the possible dependencies among the outcomes of the different auctions for users. Note that the above auction methods are described for a regular auction, where the auctioneer sells items/goods, and bidders are buyers. They can be restated for a reverse auction, where there are multiple sellers who act as bidders and the auctioneer procures items from the sellers.

Vickrey [17] has demonstrated that the bidders pay the amount of the opportunity cost for what they received as a winner, rather than the price they bid. This results from the fact that they can only determine whether they win or not. On the other hand as a bidder, the amount of their bid is used to determine their received allocation of items; however, it does not affect the amount of money they will be charged as their payment. Thus, being truthful i.e. bidding true values, becomes the (weakly) dominant strategy for the bidders.

A classical, powerful combinatorial auction mechanism is the VCG mechanism [21], named after Vickrey [17], Clarke [26] and Groves [27] who present more general versions of the previous Vickrey Auction (second-price auction) with multiple heterogeneous items. The pay price in a VCG auction is called VCG payment. VCG allocates items efficiently (maximizing social welfare), and charges bidders the cost of the items they win. Truthful bid reporting is a dominant strategy for each bidder in the VCG mechanism.

VCG auctions provide several appealing theoretical properties, and at the same time suffer from shortcomings as well. They are vulnerable to collusion by a coalition of losing bidders, the auctioneer revenue can be very low, and the process of VCG payment determination is often computationally hard. In addition, the VCG auction loses its dominant-strategy property when bidders face effective budget constraints. Therefore, the VCG auction is not often used in practice [13] [28].
2.2 Linear Program Based Algorithms

Linear and integer linear programming play an important role in algorithm design. Multiple combinatorial optimization problems can be formulated in the form of Integer Programming Problems.

2.2.1 Linear Program Formulation

A linear program (LP) is a formulation of an optimization problem (i.e., a minimization or maximization of an objective function over some domain). The objective function is linear, and is subject to linear equality/inequality constraints. Thus, linear programming is considered as one of the instances of constrained optimization. The LP model has three basic components: objective function, constraints and decision variables [1].

We use the Knapsack problem [29] as an example to illustrate a linear program. The Knapsack problem is a problem in combinatorial optimization. It derives its name from the problem someone faces when planning for a trip, and interested in filling a knapsack with items that are necessary for the trip. There are \( n \) items and each item has two attributes: a weight, and a value that quantifies the level of importance associated with each unit of that item. Since the knapsack has a limited weight capacity, the problem is to determine how many of each item to put in the knapsack so that it yields the greatest total value.

Resource allocation problems can be formulated as a Knapsack problem. The capacity of the knapsack is considered as the available amount of resources and the items as the requested resources by users. A solution to an instance of the Knapsack problem will indicate which items should be added to the knapsack. Knapsack problem can be formulated as follows:

\[
\text{Maximize } \sum_{i=1}^{n} v_i x_i \tag{2.1}
\]

\[
\text{Subject to: } \sum_{i=1}^{n} w_i x_i \leq W, x_i \geq 0
\]
The Knapsack problem is formulated as a linear program, where \( x_i, \forall i \in N \) are non-negative integer-valued decision variables, defined by \( x_i \), representing the number of type-\( i \) items that are loaded into the knapsack. The remaining terms including \( v_i, w_i \) and \( W \) are constants. A linear program consists of an objective function \( f(x) = \sum_{i=1}^{n} v_i x_i \) as a linear function to be optimized. In this function, we are given a set of items numbered from 1 to \( n \), each has a value \( v_i \) and a weight \( w_i \), and the maximum weight capacity of the knapsack is \( W \). We aim to maximize the total value of the items in the knapsack. The constraint in this problem is \( \sum_{i=1}^{n} w_i x_i \leq W, x_i \geq 0 \), which guarantees that the total weight is at most the given limit of the capacity. The left hand side of constraint \( \sum_{i=1}^{n} w_i x_i \leq W \) is a linear function while the right hand side is a constant, \( W \). Note that in a linear program, strict inequalities, i.e., “\( < \)” “\( > \)” cannot be used, because they may lead to ill-defined problems, e.g., maximize \( x \) subject to \( x < 1 \). The dual of the Knapsack problem is called dual-Knapsack which computes minimum weight needed to meet target value \( W \).

A linear program may be unbounded, infeasible, or may have a finite optimum [1]. A program is considered as an infeasible one if there is not any solution that satisfies all the constraints in the problem. The linear program can be unbounded meaning that for any feasible solution to the problem, another feasible solution with strictly higher/lower objective value can be found respectively for a maximization/minimization problem. Any solution, i.e., a setting of variables, that satisfies all the constraints is called a feasible solution. If a program is feasible and bounded then it has a finite optimum [1].

In our problem, the goal is to find the best feasible solution that maximizes total value. The 0-1 knapsack problem is the most common version of knapsack in literature [29]. In this version of the problem, the number \( x_i \) of copies of each item is restricted to zero or one, hence \( x_i \in \{0, 1\} \). The problem is called a “0-1” problem, because each item must be entirely accepted or rejected. This problem can be formulated as an Integer Linear Program. ILP is a linear program with additional constraint that some or all the variables have to
take integer values. The 0-1 knapsack optimization problem is NP-hard, while the decision problem is NP-complete; thus the optimization problem is at least as difficult as the decision problem [30].

NP-hard optimization problems do not have a structure for finding the optimal solution in an efficient way; however, it is often possible to find near-optimal solutions efficiently. LP relaxation of ILP provides a way of bounding the optimal solution [30]. Thus, the LP relaxation provides an upper bound if the problem is a packing problem (maximization optimization); otherwise, it would be a lower bound. Therefore, a feasible solution to the LP-relaxation can be considered as a fractional solution to the 0-1 knapsack problem. One of the techniques for obtaining approximation algorithms is called primal-dual schema, which will be discussed in following.

2.2.2 Constructing The Dual of a Linear Program

The original linear program of the problem is typically called the primal. For each linear program, there is another corresponding linear program called the dual. The dual problem is an LP associated to the primal. Since the dual is closely related to the primal, the optimal solution for one of them provides the optimal solution for the other one.

Fact 1. [31] Given a linear program P, if D is the dual LP of P, then P is the dual LP of D.

In order to derive the dual problem from the primal, one can use the following steps [1]:

1. Rewriting the objective function as a minimization.
2. Rewriting each inequality constraint as less than or equal and rearrange constraints in a way that the right hand side is equal to 0. (These two first steps are used to put the primal problem in a standard form.)
3. A non-negative dual variable is defined for each inequality constraint. An unrestricted dual variable is defined for each equality constraint.
4. The left-hand-side coefficients of the dual constraint are defined according to the constraint coefficients of a primal variable, and its objective coefficient defines the right-hand-side of
Table 2.1: Rules for constructing the dual program [1].

<table>
<thead>
<tr>
<th>Maximization Problem</th>
<th>Minimization Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the variable is</td>
<td>The corresponding constraint is</td>
</tr>
<tr>
<td>Unrestricted</td>
<td>⇐⇒</td>
</tr>
<tr>
<td>≤ 0</td>
<td>⇐⇒</td>
</tr>
<tr>
<td>≥ 0</td>
<td>⇐⇒</td>
</tr>
<tr>
<td>The associated variable is</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>⇐⇒</td>
</tr>
<tr>
<td>≥</td>
<td>⇐⇒</td>
</tr>
<tr>
<td>≤</td>
<td>⇐⇒</td>
</tr>
<tr>
<td>Unrestricted</td>
<td>=</td>
</tr>
<tr>
<td>≤ 0</td>
<td>≤</td>
</tr>
<tr>
<td>≥ 0</td>
<td>≥</td>
</tr>
</tbody>
</table>

the primal constraints.

5. The objective coefficients of the dual program are set equal to the right-hand-side of the primal constraints.

Considering the above steps for constructing the dual from a primal problem, the rules in Table 2.1 can be used for making the dual problem [32] [1]:

A pair of primal-dual linear programs is provided in (2.3) and (2.4) as an example. An example of constructing a dual of a primal problem is provided in Appendix A. Note that if the primal is maximization, its corresponding dual is minimization and vice versa:

Primal:

\[
Maximize \sum_{j=1}^{n} a_j x_j \quad (2.2)
\]

Subject to : \[
\sum_{j=1}^{n} c_{ij} x_j \leq b_i, i = 1, ..., m. \]

\[
x_j \geq 0, j = 1, 2, ..n. \quad (2.3)
\]
Dual:

\[
\text{Minimize } \sum_{i=1}^{m} b_i y_i \\
\text{Subject to : } \sum_{i=1}^{m} c_{ij} y_i \geq a_i, j = 1, \ldots, n.
\]

\[x_j \geq 0, j = 1, 2, \ldots, n.\]

The primal problem is in form of a maximization problem (2.3). \(x_j \forall j \in N\) are decision variables and \(b_i\) represents the total value of item \(i\) when there are \(m\) different item types.

The dual problem (2.4) is in form of minimization problem. The relationship between the objective values of primal and dual problem is provided in Table 2.2.

Primal-Dual Objective Values

Considering the primal and dual LPs defined above and Table 2.2, every feasible solution to the primal program gives a lower bound on the optimum value of the dual, and every feasible solution to the dual program gives an upper bound on the optimal value of the primal. Thus, if there is a feasible solution for the dual and the primal, then both solutions must be optimal.

The following theorem can be concluded according to the primal-dual relationship:

**Theorem 1. (LP-Duality Theorem [33]):** The primal program has finite optimum if its dual has finite optimum. In addition, if \(x = (x_1, \ldots, x_n)\) and \(y = (y_1, \ldots, y_m)\) are optimal solutions for the primal and dual programs, then \(\sum_{i=1}^{n} a_i x_i = \sum_{j=1}^{m} b_j y_j\).

Therefore, any feasible solution to the dual program gives an upper bound on the optimal value of the primal. This is the easy half of the LP-duality theorem, called weak duality which suffices for approximation algorithms. The theorem is as follows:

**Theorem 2. (Weak Duality Theorem [33])** If \(x = (x_1, \ldots, x_n)\) and \(y = (y_1, \ldots, y_m)\) are feasible solutions for the primal and dual programs, then \(\sum_{i=1}^{n} a_i x_i \leq \sum_{j=1}^{m} b_j y_j\)
Table 2.2: Possible combinations for the primal and its dual [1].

<table>
<thead>
<tr>
<th></th>
<th>Finite Optimum</th>
<th>Unbounded</th>
<th>In-feasible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite Optimum</td>
<td>Possible</td>
<td>Impossible</td>
<td>Impossible</td>
</tr>
<tr>
<td>Unbounded</td>
<td>Impossible</td>
<td>Impossible</td>
<td>Possible</td>
</tr>
<tr>
<td>In-feasible</td>
<td>Impossible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
</tbody>
</table>

2.3 Resource Allocation in Cloud Computing

Cloud computing is a service to render computational resources over the Internet. One of the key features of cloud computing is the availability of computing resources that can be requested by users on demand. Thus, cloud providers need to be capable of allocating resources according to users’ requests. Resource Allocation (RA) in cloud computing environment is defined as the procedure of allocating computational resources and cloud computing services among cloud users according to their demand requests.

The amount of resources in a cloud provider is plentiful, but this amount is not infinite. Thus, if resources in cloud servers are not managed or allocated properly, there will be multiple challenges such as the problem of resource starvation. Generally it happens because of under/over provisioning cloud resources among cloud users. The problem can be addressed by resource provisioning methods, which help cloud providers to manage resources for each set of users [34]. Moreover, it is needed to deal with resource heterogeneity in cloud systems. Besides, resource allocation techniques should be aware of the available capacity and status of each resource in the cloud system; this means that cloud system providers should use this information efficiently and apply mechanisms to deal with heterogeneous users’ demands over time by resource allocation methods.

There are multiple resource allocation strategies providing ways of integrating activities of the cloud system provider in order to have an efficient allocation of scarce resources. These methods need information about the system such as type of resources, the amount of resources and the time of request. According to [34] [35], an optimal resource allocation method avoids resource contention, which is defined as accessing the same resource simultaneously.
by multiple users, under/over provisioning of resources and resource fragmentation.

Predicting the demands of users and their applications in advance is considered as a challenge for cloud providers. Therefore, efficient resource allocation methods are needed to be able to deliver the resources to the users efficiently, considering heterogeneity and limited resources. Scheduling algorithms and online auction methods are considered as useful methods for effective resource allocation.

### 2.4 Pricing Models

After allocation, the auctioneer needs to determine payment for the allocated item/bundle of items, from each winning bidder. These prices can also be used as a guideline for future auctions. The term *price* has traditionally been used for constructing artificial distributions to define the amount offered for a bundle of resources. However, this term refers to the amount a bidder has decided to pay for a bundle. This value is mechanism-specific and it is not necessarily the same as the amount offered [36]. Thus, different pricing models need to be defined according to the auction system properties.

New pricing methods have been emerging, including multiple factors for calculating prices. These methods result in more flexible pricing models than traditional methods. Each cloud provider has its own pricing method. For instance, a typical pricing approach is to pay once for unlimited resource usage. However, this approach is rigid and does not consider other factors related to pricing such as price fairness [8]. Amazon Web Service [37] and Google App Engine [38] use *pay-per-use fixed pricing*. Users are charged based on their overall resource consumption. Multiple other cloud providers use *pay for the resources* method, where the payments are calculated based on the amount of storage or bandwidth size [39]. A prevalent pricing scheme in cloud centers is the *pay-per-use* method, which is based on units with constant price [40]. Another prevalent method is known as the *subscription method*, where users pay a constant price for a specific service unit and period of time. Providers
prefer to use a static flat-rate pricing schema. However, dynamic pricing results in more efficient schema in the case of providing high-value services and resources [39].

In the cloud computing literature, cloud resource pricing has been widely studied. However, most models assume the pay-per-use pricing method. Similar to other infrastructures or IT services, pricing mechanisms are considered as an important issue for a cloud computing infrastructure, thus appropriate pricing models can result in success of cloud services [34]. Osterwalder et al. [2] state that earlier pricing methods are quite operational; however, regarding the changing scenario of the current resource allocation mechanisms and unpredictable environment of demand and supply services, strategic planning should be considered.

Three basic types of pricing methods for cloud services are summarized in Table 2.3 [2]. Note that in fixed pricing models, the prices do not change for different scenarios. Thus, it is regardless of the market environment and criteria. Differential pricing methods consider the user and product features and other parameters related to user’s choice, but do not consider the real market scenario. Market pricing methods offer prices based on real market conditions [34]. New cloud providers apply auction-based pricing models for their resource allocation and charging system. Using auction-based models provides multiple advantages, including higher revenue for providers and rational prices for users. Moreover, price changing is tightly correlated with fluctuation in demand-supply.

2.5 Simulations for Auction Methods

Simulation studies are considered as a useful structure for evaluating the performance of auction methods. Generating artificial data to represent different scenarios for the auctioneer is necessary in measuring the efficiency of the auction method. In studies by Fujishima et al. [15] and Boutilier et al. [41], multiple bid generation techniques are suggested based on criteria such as number of items and bids. However, Leyton-Brown et al. studied drawbacks of these techniques in [36], including:
1) Most data generation techniques assume equal number of items to be requested in a bundle of same size.

2) These data generation techniques determine the number of items to be in a bundle regardless of which items have already been chosen for the bundle.

3) These methods use prices that are randomly chosen from various intervals such as: interval $[0, 1]$, normal distribution with a specific mean and standard deviation, interval $[0, r]$ where $r$ is the number of requested items, etc. Using an interval $[0, 1]$ or a normal distribution has a disadvantage of no dependency to the number of items. Moreover, using the interval $[0, 1]$ does not consider which items have been requested in the bundle. In the method based on interval $[0, r]$, mean and range are parameterized by similar variables.

A Combinatorial Auction Test Suite (CATS) is provided by Leyton-Brown et al. [36]. This tool models a realistic bidding behavior using a set of distributions, providing a useful tool to study online auction performance. In this method, bids are generated from a graph that depicts the economic relationships among items. In this method, the number of items included in each bundle relates to which items it contains. In addition, specific items are more likely to appear together when they are complementary. Therefore, when bids are generated, the prices offered are related to items included in the bundle. However, this suite is designed for single-unit combinatorial auctions in which all goods are distinguishable from each other. In our model, we have multiple units of different resource types that can be requested by users in the auction. For experimental studies, we use synthesized data to evaluate our method. However, we adopt the distribution and bid generation technique [36] to generate combinations of resource requests and a value associated to each bid to describe the price.
Table 2.3: Different pricing models for cloud computing infrastructure [2].

<table>
<thead>
<tr>
<th>Category</th>
<th>Pricing Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Prices</td>
<td>Pay per use</td>
<td>User pays according to her usage.</td>
</tr>
<tr>
<td></td>
<td>Menu Price</td>
<td>Fixed price mentioned in catalogs.</td>
</tr>
<tr>
<td></td>
<td>Subscription</td>
<td>Users pay flat prices to receive the resource.</td>
</tr>
<tr>
<td>Market Pricing</td>
<td>Dynamic Market</td>
<td>Large number of seller and buyers are present which influences pricing criteria.</td>
</tr>
<tr>
<td></td>
<td>Auction</td>
<td>Users bid increasingly.</td>
</tr>
<tr>
<td></td>
<td>Reverse Auction</td>
<td>Sellers bid decreasingly in price values.</td>
</tr>
<tr>
<td>Differential Pricing</td>
<td>Service Based</td>
<td>Charging price is based on service feature.</td>
</tr>
<tr>
<td></td>
<td>Value based</td>
<td>Charging price is based on user’s value proportion.</td>
</tr>
<tr>
<td></td>
<td>Volume based</td>
<td>Charging price is based on volume of resource consumed.</td>
</tr>
</tbody>
</table>
Chapter 3

Related Work

Cloud computing systems offer enormous potential for cost-effective and flexible computing services. These systems present a different way to manage computing resources, which motivated academic research, published white papers and surveys such as \cite{42} \cite{43} \cite{44} \cite{45}.

Endo \textit{et al.} discuss the cloud computing market, its challenges in resource sharing, and review the open-source solutions for cloud computing \cite{42}. Voorsluys \textit{et al.} \cite{45} provided a survey on cloud computing fundamentals. First, an introduction on cloud computing is provided. Then, layers and types of clouds are discussed and desired features of a cloud market are presented. Later, managerial tasks in cloud systems are covered. Resource allocation is mentioned as one of the aspects of these managerial tasks.

Resource allocation is considered as a classic problem in computing systems that has been studied extensively. These studies also include approaches of designing pricing mechanisms that maximize the social welfare and approaches of incentive compatible allocations. Therefore, optimal pricing of the resources has become a major challenge for the cloud service providers.

Multiple pricing models for cloud computing systems are discussed by Kansal \textit{et al.} \cite{46}. The discussed models are mostly related to fixed-price methodologies. Our work aims at the usage-based pricing functions for multi-resource cloud centers, towards more efficient cloud systems.

Since cloud computing deals with multiple kinds of virtual resources, scheduling plays an important role in cloud computing. Moreover, regarding the large number of unit resources in these systems, manual scheduling is not a feasible solution. Therefore, there is a need to design different types of scheduling algorithms for these systems. These schedul-
ing algorithms can result in efficient resource allocation in cloud systems [43]. A survey on scheduling algorithms in cloud computing and grids is presented by Shenai et al. [43]. In this work, multiple existing scheduling algorithms are reviewed, and are categorized based on scheduling parameters such as execution time, response time, cost, scalability of the algorithm resources, and trust.

Earlier works on deadline-sensitive job scheduling with partial execution and non-linear utility functions have been published by Zheng et al. [47] and He et al. [48]. Zheng et al. [47] focused on the preemptive scenario, in which a job in service can be interrupted by other jobs, and can be resumed later. Then, the scheduling mechanism is proved to be able to achieve maximum utility in case of jobs with homogeneous deadlines. For jobs with heterogeneous deadlines, it is proved that their mechanism achieved a competitive ratio greater than 2. However, all these works consider one resource (CPU only) and don’t cover the case of multi-resource sharing.

Previous works on multi-resource allocation mainly focus on the offline setting [49], [50]. For instance, Jain et al. provide a scheduling mechanism based on greedy algorithms for multi-resource sharing to users with deadline-sensitive jobs in computing clusters [51]. Users are charged for their received resources by marginal values according to the properties of the model. However, the proposed mechanism works in an offline manner, assuming that all information about users are known before running the allocation and payment rules.

Auctions have long been used as a mechanism for allocating resources to competing users. As well as individuals and private organizations, the public sector also benefits from auctions in transferring resources to multiple users. With access to the Internet, the implementation of auctions has witnessed online deployment. These developments result in the need of mechanisms to improve the efficiency and trustworthiness of these systems. Moreover, auctions are considered as mechanisms that combine allocation and pricing approaches, for pursuit of economic efficiency and truthfulness [52]. Thus, auction mechanisms have been studied
as an efficient market mechanism design in cloud computing market \cite{3,53,54}. However, there are multiple challenges in implementing these mechanisms; much research has been done to address some of the challenges faced by the auctioneer and/or bidders when implementing these systems. One of these challenges is providing a truthful auction mechanism. The celebrated VCG auction \cite{17,27} is a well-known type of auction that provides a truthful mechanism in which bidders have no incentive to report a falsified bid. It is the only type of auction mechanism that is able to guarantee the truthfulness and absolute social welfare maximization simultaneously, using an optimal allocation rule and a VCG pricing function \cite{52}. However, the allocation problem is often NP-hard, and is computationally infeasible to solve at large scale \cite{27}.

A crucial concern about auctions in practice is the ability of users to collude. Bidding collusion happens when some or even all of the users constitute a bidding ring and significantly reduce their bid values. Then, the revenue of the provider will get reduced. Wang et al. \cite{55} present a mechanism for VM allocation, which is collusion resistant. However, the mechanism considers only one-round auctions and has a relatively large competitive ratio of $O(\sqrt{n})$, where $n$ is the number of VM instances.

Dynamic resource provisioning using auction mechanisms has been studied in the past few years. Extending single-round auctions into online auctions usually compromises truthfulness due to the lack of information about the future; this needs to be handled using methods such as the supply curve \cite{56,57}. Zhang et al. \cite{54} studied the resource allocation problem in cloud computing where the demand arrivals are real time, and they presented a truthful online auction for resource allocation. They use a well-known technique for achieving truthfulness in online auctions based on the concept of supply curves. The online auction is able to serve the heterogenous demands of users. However, a single type of VMs is considered. Zaman et al. \cite{58} considered dynamic resource provisioning and present a truthful mechanism that can improve the utilization and increase the efficiency of resource allocation. However, the
social welfare approximation ratio is not guaranteed in their work.

Mashayekhy et al. [59] proposed a cloud reward-based scheduling for resource allocation. They considered an environment for reward-based scheduling when there are periodic tasks and execution of each task consists of a mandatory and an optional part. If the processor schedules all the mandatory parts of a task, the task will obtain a value. It will receive an additional reward value if its optional executions are scheduled by the processor as well. The proposed scheduling mechanism is truthful, but without proven guarantee on its competitive ratio. Azar et al. [60] provide a truthful online mechanism for resource allocation with commitments with a well-designed payment rule. The mechanism is designed for preemptive scheduling with deadlines, and the main goal is to maximize the total value of completed jobs. Although the presented scheduling mechanism provides a constant competitive ratio, that ratio is relatively large. Moreover, they show that the mechanism can satisfy the commitment property, only for large value of deadline slackness. Deadline slackness is defined as the ratio of temporal constraint, user’s deadline, to total processing time in the system, T.

An online multi-resource sharing problem is presented by Kash et al. [61]. They used the fair division theory to allocate multiple computational resources among users. They considered the dynamic model of the fair division, and proposed desirable properties for dynamic resource provisioning. However, it is assumed that once a user arrives to the system, it will never leave the system, so there is no deadline constraint. Mashayekhy et al. [62] proposed an online auction mechanism for resource allocation and pricing in cloud systems. The proposed online mechanism makes no assumptions about the future demand of resources, and allocates resources to the selected users for the period they are requested. The mechanism is proven to be incentive-compatible with a well-designed payment function based on a factor named bid density. However, there is no approximation ratio proven.

Zhang et al. [7] designed a randomized auction for dynamic resource allocation in cloud computing systems, which is truthful based on a convex decomposition method. It guarantees
a small approximation ratio for social welfare. Shi et al. [52] studied dynamic resource allocation where cloud users are subject to budget constraints. Considering users’ fixed budgets, they presented an online auction where decisions are made based on time domain and limited budget. However, the work does not include the temporal correlation in decision making since jobs may include multiple time slots. Zhang et al. [53] studied online cloud auctions in which users bid for a fixed time window for job execution, which eliminates the scheduling dimension. Zhou et al. [3] studied the cloud market for jobs with different completion deadlines. They use the posted pricing method to design an online mechanism for resource provisioning for tasks with deadlines, and proved a reasonable competitive ratio.
Chapter 4

Pricing Functions for Cloud Auctions

In this chapter, we first describe the system model for our cloud computing system. First, we review the system model properties and bidding language required to run the auction mechanism in the system in Section 4.1. The optimization problem and system constraints are covered in this section as well. Then, in Section 4.2, the system scheduling algorithm is discussed. Later, in section 4.3, we propose three marginal pricing functions that are designed according to the system’s criteria and properties. The fixed-price technique is also covered in this section as one of the typical basic pricing methods for providers in the cloud market. Further along the chapter, we provide the theoretical analysis for the online auction mechanism using our proposed pricing methods. First, the computational complexity of the mechanism is discussed. Next, the competitive ratio of the mechanism with one of our proposed pricing functions is evaluated and at the end of the chapter, truthfulness of the mechanism is discussed.

4.1 Our Cloud Auction System and Bidding Language

We consider a cloud provider who acts as the auctioneer, providing a pool of \( R \) types of multiple resources such as RAM, CPU and storage to multiple cloud users \( I = \{1, 2, 3, ..., 100\} \) through an online auction mechanism. Assume that type \( r \) has total \( C_r \) units of resource, which is a non-negative number. The resources will be scheduled over a finite time interval \( T = \{1, 2, ..., 24\} \). Let \( a_r(t) \) be the amount of type\( -r \) resource allocated at time \( t \) in the cloud auction system, where \( a_r(t) = 0 \) at the beginning.

Cloud users arrive in an arbitrary manner, and request resources by offering bids to the auctioneer. Note that multiple bids can arrive simultaneously. User \( i \) presents her bid \( B_i \)
whenever she arrives to the system. Requests will then be processed by the auctioneer. In this model, each user requests different amounts of multiple types of resources, while stating a deadline for receiving desired resources. Thus, user $i$’s bid would be expressed as follows:

$$B_i = (a_i, d_i, l_i, b_i, \{c_i^r\}_{r \in R})$$  \hspace{1cm} (4.1)

Here, $B_i$ is the offering bid bundle of user $i$, and $a_i$ is the arrival time of user $i$, $d_i$ is user $i$’s deadline for job completion. Let $c_i^r$ be the desired amount of type-$r$ resource for user $i$, and $l_i$ be the number of time slots required to finish the job by the deadline. Note that user $i$’s job can be executed at any time slot and the job execution doesn’t need to be continuous, as long as the total execution meets the deadline. Let $b_i$ be the corresponding bidding price if the desired job is completed before deadline $d_i$, while $v_i$ is the true valuation of user $i$’s bid. Note that the valuation is known only to the bidder herself. We adopt the XOR bidding rule, such that a user can win at most one bid among all her optional bids. At the time of bid arrival, the cloud provider decides immediately whether to accept the bid or not. Let the binary variable $x_i$ represent the auction decision:

$$x_i = \begin{cases} 
1 & \text{if user } i \text{'s bid is accepted} \\
0 & \text{otherwise} 
\end{cases} \hspace{1cm} (4.2)$$

Define another scheduling binary variable $y_{i,t}$ as follows:

$$y_{i,t} = \begin{cases} 
1 & \text{if user } i \text{'s job is scheduled to run at time } t \\
0 & \text{otherwise} 
\end{cases} \hspace{1cm} (4.3)$$

representing the schedule of user $i$’s job. Note that once a user’s bid is accepted, the bid will be fulfilled; the user receives the total requested amount of resources and number of time slots before her deadline. Since the mechanism is online, the auctioneer makes the decisions in an online manner and allocates the resources instantly. Thus, it is not possible for users to change their schedule or cancel their requests. The auctioneer also determines the payment
for each user $i$ as $p_i$. Therefore, the utility of user $i$ is:

\[
u_i(b_i) = \begin{cases} 
v_i - p_i, & \text{if } x_i = 1 \\
0, & \text{if } x_i = 0
\end{cases}
\]

(4.4)

User $i$’s utility refers to the net profit user $i$ gets from receiving the allocated resources. Note that the bidders are assumed to be selfish and rational, trying to maximize their utility. The cloud provider, instead, aims to achieve the highest social welfare.

**Definition 1. (Social Welfare).** The social welfare in the presented cloud auction is the aggregate user utility $\sum_{i \in [t]} v_i x_i - \sum_{i \in [t]} p_i$ plus the cloud provider’s utility $\sum_{i \in [t]} p_i$.

It can be seen that payments cancel out each other. Thus, the social welfare will be $\sum_{i \in [t]} v_i x_i$.

In Definition 1, $\sum_{i \in [t]} p_i$ is the summation of the payments for all the users accepted to the auction mechanism, who received the requested resources and are charged $p_i$. Therefore, $\sum_{i \in [t]} p_i$ is the total revenue, that the provider receives from winners of the auction.

The provider decides two main parts of the auction mechanism: allocation rule and payment rule.

1) Allocation: The provider decides whether to accept the users’ bids or not, and if accepted how to schedule the job. For this section, we adopted the deadline-sensitive scheduling algorithm proposed by Zhou et al.\[3\] and modified it to be applicable to our system model. More information is provided in Sections 4.1.1 and 4.2.

2) Payment: Next, a payment rule is used for the auction mechanism, which is determined according to the users’ bids and works in an online manner. For this part, we propose three different marginal pricing functions in accordance to our system model properties. The pricing functions are discussed in detail in Section 4.3.

4.1.1 The Online Auction Problem

Social welfare is a commonly used criteria for evaluating the outcome of an auction mechanism\[63\]. Recall that social welfare for our problem can be considered as the summation
of user valuations $E = \sum_{i \in I} v_i x_i$. At the time of resource allocation, we wish to make sure that the allocation method respects the capacity constraints for each type of resource in the cloud. Thus, the finite supply of each resource type can be formulated into the following capacity constraint, which guarantees that the allocated resources never exceed the total available resource capacity. Note that $C_r$ is a fixed value for each type-$r$ resource.

$$\sum_{i \in [I]: a_i < t} c_i^r y_{i,t} \leq C_r, \quad \forall r \in [R], \forall t \in [T]. \quad (4.5)$$

Moreover, the job should be scheduled to run between its arrival time and its desired deadline, resulting in the following constraint:

$$y_{i,t} \leq d_i x_i, \quad \forall t \in [T], \forall i \in [I], a_i \leq t. \quad (4.6)$$

In addition, the number of allocated time slots should be no fewer than the requested time slots by user $i$ for job completion:

$$l_i x_i \leq \sum_{t \in [T]: t \geq a_i} y_{i,t}, \quad \forall i \in [I] \quad (4.7)$$

Constraint (4.7) is for the case when user $i$’s job is not scheduled to run at time $t$, $y_{i,t} = 0$. If $y_{i,t} = 0$, constraints (4.5) and (4.6) hold. However, the user did not receive any resources for time $t$ and should not be considered as a winner. For $y_{i,t} = 0$ constraint (4.7) does not hold, so the user is not in the set of winners.

Considering the above constraints for our cloud auction model, the social welfare maximization problem, under truthful bidding $b_i = v_i$, is formulated as:

$$\text{Maximize } \sum_{i \in [I]} b_i x_i \quad (4.8)$$
Subject to:

\[ \sum_{i \in [I]: a_i \leq t} c_i y_{i,t} \leq C_r, \forall r \in [R], \forall t \in [T] \]  

(4.8a)

\[ y_{i,t} \leq d_i x_i, \ \forall t \in [T] : a_i \leq t, \forall i \in [I] \]  

(4.8b)

\[ l_i x_i \leq \sum_{t \geq a_i} y_{i,t}, \forall i \in [I] \]  

(4.8c)

\[ x_i, y_{i,t} \in \{0, 1\}, \forall t \in [T], \forall i \in [I] \]  

(4.8d)

Note that constraint (4.8a) ensures the capacity limit of each type of resource. Constraint (4.8b) guarantees that a job is scheduled between its arrival and deadline. Having a sufficient number of allocated time slots for successful service delivery is represented as constraint (4.8c). Constraint (4.8d) is added to ensure that XOR bidding rule is satisfied. Using XOR bidding rule, a user can win at most one bid among all her submitted bids.

In literature, the classic optimization problem below, 0-1 Knapsack, is proven to be NP-hard. The proof is provided by Kellerer et al. in [29].

\[
\text{Maximize } \sum_{i \in [I]} v_i x_i \\
\text{Subject to } \sum_{i \in [I]} w_i x_i \leq W, x_i \in \{0, 1\}
\]

Since the classical NP-hard knapsack problem is a special case of the problem in ILP (4.8), the cloud auction problem is also NP-hard. Thus, we resort to an efficient approximate solution algorithm.

Note that even the offline version of ILP (4.8) without deadline and time slots constraints, (4.8b) and (4.8c), is still NP-hard. Thus, we need to address these challenges and reformulate ILP (4.8). In order to address the challenge of considering deadline constraints, we adopt the compact-exponential ILP with a packing structure and reformulate the original ILP (4.8) into a simplified version that addresses this challenge, but involves an exponential number of variables. Therefore, ILP (4.8) is reformulated as:
Maximize \( \sum_{i \in [I]} \sum_{k \in \Lambda_i} b_{ik} x_{ik} \)  \hspace{1cm} (4.9)

Subject to:

\[
\sum_{i \in [I]} \sum_{k \in \Lambda_i} c_{i}^r x_{ik} \leq C_r, \forall r \in [R], \forall t \in [T] \hspace{1cm} (4.9a)
\]

\[x_{ik} \in \{0, 1\}, \forall i \in [I], \forall k \in \Lambda_i \hspace{1cm} (4.9b)\]

\[\sum_{k \in \Lambda_i} x_{ik} \leq 1, \forall i \in [I] \hspace{1cm} (4.9c)\]

Constraint (4.9a) is equivalent to (4.8a), and \( \Lambda_i \) is the set of time schedules that are able to satisfy constraints (4.8b) and (4.8c). The number of variables in (4.9) is exponential because the number of possible time schedules for user \( i \) will be exponential in size. Note that the value of \( b_{ik} \) is based on schedule \( k \) and is equal to the corresponding \( b_i \). Later, constraint (4.9b) will be relaxed to \( x_{ik} \geq 0 \).

Next, to solve this problem, we resort to the primal-dual algorithm design technique. We construct the dual problem of ILP (4.9). Following the instructions that are provided in Chapter 2 for deriving the dual of a primal program, we define dual variables \( p_r(t) \) for constraint (4.9a) and \( u_i \) for constraint (4.9c). Using dual variables \( u_i \) and \( p_r(t) \), the dual problem is formulated as follows. Table 4.1 provides notation for ease of reference.

Minimize \( \sum_{t \in [T]} \sum_{r \in [R]} C_r p_r(t) + \sum_{i \in [I]} u_i \)  \hspace{1cm} (4.10)

Subject to:

\[u_i \geq b_{ik} - \sum_{r \in [R]} \sum_{t \in [k]} c_{i}^r p_r(t), \forall i \in [I], \forall k \in \Lambda_i \hspace{1cm} (4.10a)\]

\[u_i, p_r(t) \geq 0, \forall i \in [I], \forall r \in [R], \forall t \in [T] \hspace{1cm} (4.10b)\]

Note that a solution to ILP (4.9) is a corresponding feasible solution to the main optimization problem (4.8). Recall that the number of variables in ILP (4.9) is exponential,
Table 4.1: Summary of notation in cloud auction system model

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_r$</td>
<td>Total capacity of type-$r$ resource</td>
</tr>
<tr>
<td>$I$</td>
<td>Number of users</td>
</tr>
<tr>
<td>$p_i$</td>
<td>User $i$’s payment</td>
</tr>
<tr>
<td>$[X]$</td>
<td>Integer set ${1, \ldots, X}$</td>
</tr>
<tr>
<td>$T$</td>
<td>Number of time slots</td>
</tr>
<tr>
<td>$B_i$</td>
<td>User $i$’s bid bundle</td>
</tr>
<tr>
<td>$a_i$</td>
<td>Arrival time of user $i$</td>
</tr>
<tr>
<td>$d_i$</td>
<td>Deadline of job completion for user $i$</td>
</tr>
<tr>
<td>$b_i$</td>
<td>User $i$’s bidding price</td>
</tr>
<tr>
<td>$c_i^r$</td>
<td>Demand of type-$r$ resource by user $i$, $\forall r \in [R]$</td>
</tr>
<tr>
<td>$v_i$</td>
<td>True valuation of user $i$’s bid</td>
</tr>
<tr>
<td>$x_i$</td>
<td>Equals 1 when $i$ wins; otherwise 0</td>
</tr>
<tr>
<td>$y_{i,t}$</td>
<td>Whether or not to allocate user $i$’s job in slot $t$</td>
</tr>
<tr>
<td>$a_r(t)$</td>
<td>The amount of allocated type-$r$ resource at time $t$</td>
</tr>
<tr>
<td>$p_r(t)$</td>
<td>Marginal price of type-$r$ resource at time $t$</td>
</tr>
<tr>
<td>$U_r$</td>
<td>Maximum value per unit of type-$r$ resource per unit of time</td>
</tr>
<tr>
<td>$L_r$</td>
<td>Minimum value per unit of type-$r$ resource per unit of time</td>
</tr>
<tr>
<td>$u_i$</td>
<td>Utility of user $i$</td>
</tr>
<tr>
<td>$l_i$</td>
<td>Number of time slots requested by user $i$</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>A positive parameter such that $\sigma = \frac{T}{\min_{i \in [I]} l_i}$</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>Competitive ratio of Algorithm 1</td>
</tr>
</tbody>
</table>
since the number of possible time schedules for user $i$ is exponential in size. However, using a primal-dual allocation scheme provides a solution that only updates a polynomial number of variables, as discussed in the next section. This method enables us to solve the optimization problems (4.9) and (4.10).

### 4.2 Online Auction Design

In the allocation mechanism, the provider as the auctioneer needs to decide whether to accept user $i$’s job or not, and if yes, how to schedule her requested job in a way to meet the deadline. In order to solve the primal problem, we apply the primal-dual technique to the primal (4.9) and its dual (4.10). Complementary Slackness [64] provides a relation between primal problem and dual problem. According to complementary slackness, the number of variables in the dual problem is equal to the number of constraints in the primal and the number of constraints in the dual is equal to the number of variables in the primal. Therefore, variables in one problem are complementary to constraints in the other. Thus, for each primal variable $x_{ik}$, there is a dual constraint associated to it and update of a primal variable is based on its dual constraints. $x_{ik}$ is zero unless its associated dual constraint (4.10a) is tight, so the dual variable $u_i \geq 0$. Dual variable $u_i$, can be defined as the maximum of zero and right hand side of (4.10a) as follows:

$$u_i = \max\left\{0, \max_{k \in \Lambda_i} \{b_{ik} - \sum_{r \in [R]} \sum_{t \in k} c_r^i p_r(t)\}\right\}$$

(4.11)

This formulation is concluded from $u_i > 0$ and the right hand side of (4.10b), and it can be seen that it leads to user $i$’s utility maximization. Therefore, user $i$ will be accepted if $u_i \geq 0$ and its job will be served according to the schedule that maximizes $b_{ik} - \sum_{r \in [R]} \sum_{t \in k} c_r^i p_r(t)$. Otherwise, in case of $u_i < 0$, the user will be rejected. Assignment of $u_i$ in equation 4.11 maximizes user $i$’s utility, resulting in social welfare maximization. Note that different users can have different schedule $k$ with different possible time slots.
The size of the dual constraint (4.10a) is exponential because the user requests $l_i$ time slots among total $T$ time slot, considering their arrival time and deadlines. $(d_i - a_i)$ is exponential in number of time slots. Although the size of dual constraint (4.10a) is exponential, a polynomial number of dual constraints can be selected using a dual oracle. The dual oracle contains the following steps. First, for each user $i$’s job, $l_i$ time slots are selected with the minimum price for $t \in [a_i, d_i]$. Next, let $k$ be the corresponding schedule, which is added to the set $K_i$. The schedule that maximizes user $i$’s utility is the schedule with the minimum price in set $K_i$.

Next, we update the dual variable $p_r(t)$. Recall that $a_r(t)$ is defined as the amount of allocated resource type-$r$ at time $t$, while $p_r(t)$ is defined as the marginal price per unit of type-$r$ resource at time $t$. Thus increasing the value of $a_r(t)$ will result in an increase in value of $p_r(t)$, since it is a function of $a_r(t)$. Moreover, in case that user $i$ wins, $\sum_{t \in k} \sum_{r \in [R]} c_i^r p_r(t)$ denotes the total charge for user $i$ when her job is assigned according to schedule $k$. Thus, we need to design a marginal price function according to the model properties.

In order to design an efficient pricing function, we try to define the function depending on the amount of resources that has been sold and user’s demand, rather than defining the price function according to users’ bid values. The main reason for following this approach is the fact that if the price function depends on users’ bid values, it motivates users to falsify their bids so they can affect their payment. On the other hand, if the pricing function depends on user’s demand instead of bid value, users’ payment would not be dependent on users bid values. We first define the following minimum and maximum values to be used in designing pricing functions.

We define the maximum and minimum values per unit of type-$r$ resource for each unit of time as $U_r$ and $L_r$ respectively. Specifically $U_r = \max_{i \in [I] : c_i^r > 0} \{ b_i / c_i^r \}$ and $L_r = \min_{i \in [I]} \{ t \sum_{r \in [R]} \frac{b_i}{\sum_{j \in [I]} c_j^r} \}$. Note that the $U_r$ and $L_r$ are defined according to the system model and properties. $U_r = \max_{i \in [I] : c_i^r > 0} \{ b_i / c_i^r \}$ is defined considering constraint (4.10a), where if
there is no type $r$ resource available, then that type of resource cannot be allocated to a user. In such a case, the user’s request is rejected and her utility is zero. We then need to make sure that the right hand side of constraint (4.10a) is zero as well. The value of $L_r$ is also defined in accordance with the initial objective value of the dual problem (4.10) computed by the online algorithm defined in the following. The initial objective value of the dual problem is $D_0 = \sum_{t \in T} \sum_{r \in R} C_r p_r(t) = \sum_{t \in T} \sum_{r \in R} C_r \frac{L_r}{e^{R \sigma}}$. Thus, defining $L_r = \min_{i \in I} \left\{ \frac{b_r}{\sum_{r \in R} c_{r|i}} \right\}$ satisfies this formulation of initial value.

**Algorithm 1** Primal Dual Online Auction Mechanism $PD_1$

1: **Input**: $C_r$ and bidding bundle $B_i$
2: DEFINE $p_r(a_r(t))$ according to one of the proposed pricing functions
3: INITIALIZE: $y_{i,t} = 0, a_r(t) = 0, x_i = 0, p_r(t) = 0, u_i = 0, \forall i \in [I], \forall r \in R, \forall t \in T$. and $x_{ik} = 0, \forall i \in [I], k \in \Lambda_i$
4: Upon the arrival of ith user
5: Get bidding language $B_i$ according to [4.1]
6: $(x_i, \{y_{i,t}\}, p_i, \{p_r(t)\}, \{a_r(t)\}) = A_{schedule}(B_i, \{C_r\}, \{p_r(t)\}, \{a_r(t)\})$.
7: **if** $x_i = 1$ **then**
8: User $i$’s bid is accepted and resources are allocated according to $y_{i,t}$. Charge user $i$ by price $p_i$.
9: **else**
10: Reject the ith bid.
11: **end if**

Then, the pricing function $p_r(t)$ should be defined so that it starts at a value less than or equal to $L_r$ ($0 \leq p_r(t) \leq L_r$), and increases as the value of allocated resources at time $t$, $a_r(t)$, increases. Then at the time that $a_r(t) = C_r$ the value of $p_r(t)$ should reach to the upper bound value. So it should be at least $U_r$ ($p_r(t) \geq U_r$), because in this case, the cloud provider would not allocate any more resource of type-$r$ to any user (the allocated resources of type-$r$ reached its total capacity). Considering the definitions and properties discussed above, multiple marginal payment functions are proposed in the next section.

Auction Mechanism

Guided by the discussions above, the online auction algorithm $PD_1$ is designed as given in online mechanism $PD1$. Algorithm 1 first selects the pricing function in line 2, and initializes
Algorithm 2 Scheduling Algorithm $A_{\text{schedule}}$

1: **Input:** $\{C_r\}, B_i, \{p_r(t)\}, \{a_r(t)\}$
2: **Output:** $\{a_r(t)\}, \{p_r(t)\}, p_i, x_{ik}$
3: $c(t) = \sum_{r \in R} c^r_i p_r(t), \forall t \in T.$ // price per slot
4: Select $l_i$ slots with minimum value of $(c(t))$ while $a_r(t) + c^r_i \leq C_r, \forall r \in R,$ in $[a_i, d_i],$ add the schedule to $k.$
5: $p_i = \sum_{t \in L} c(t)$
6: $u_i = b_i - p_i$
7: **if** $u_i > 0$ **then**
8: $x_{ik} = 1, y_{i,t} = 1, \forall t \in k.$
9: $a_r(t) = a_r(t) + c^r_i, \forall r \in [R], t \in k.$
10: Update the dual marginal price variables
11: $p_r(t) = p_r(a_r(t)), \forall r \in [R], t \in k$
12: **else**
13: $x_{ik} = 0;$
14: **end if**
15: **Return** $x_{ik}, \{y_{i,t}\}, \{a_r(t)\}, \{p_r(t)\}, p_i,$

the primal and dual variables in line 3. Upon the arrival of each user $i$, we select the time slot with minimum value of $c(t)$ through the dual oracle (lines 4-6). If user $i$’s utility was positive in line 7, primal variables are updated in line 8. Then the usage of different resources are increased by $a_r(t)$ and the price is also updated in lines 9 and 11 in $A_{\text{schedule}}$. At the end, the auction decision is announced in lines 6-11 in $PD_1$.

4.3 Proposed Pricing Functions

Charging a user based on how much she uses the service is customary in many businesses, including cloud-based infrastructures. One of the reasons for choosing a usage-based method is the fact that these pricing methods give providers a tool to manage traffic. They result in traffic smoothing by varying the price of service with time. Moreover, these pricing methods reduce the degree of overload for the provider, since users can be encouraged to alter their usage pattern. Thus, the load of the network will be shifted. In addition, usage-based pricing is consistent with a temporary need for reserved end-to-end bandwidth for certain high-volume real-time data applications. Therefore, the usage-based pricing method
is adopted by multiple cloud-based systems [65] [66] [67].

In this section, we propose multiple marginal pricing functions based on the amount of resource that has been sold and user’s demand. Then, we examine the performance of different functions for the presented auction mechanism and evaluate pricing functions for resource allocation with deadlines. As was discussed in the previous section, for designing the pricing functions for our auction model, multiple facts need to be considered. First, the price value is defined as the unit price for each type of resource, i.e. the pricing method is a function of allocated resource of type-\(r\) at time \(t\), \(a_r(t)\) denoted as \(p_r(a_r(t))\). Second, the value of the price should be in accordance with the definition of \(L_r\) and \(U_r\). The price value starts at \(L_r\) or below and increases during the allocation of resource type-\(r\) until it reaches \(U_r\) or exceeds. In practice, the value of unit price can be smaller than \(L_r\), such as starting at 0. The same for the upper bound \(U_r\) — the value of the unit price can be larger than \(U_r\). The point is that the unit price acts as a non-decreasing function between these two bounding points.

Moreover, the pricing function should be in accordance to the allocation rule. In this case, at the beginning it is assumed that \(a_r(0) = 0\). Thus, the corresponding price value should be in accordance with allocating zero amount of resource type-\(r\). At the end of the allocation process, the value of \(a_r(t)\) will reach \(C_r\). Thus, the unit price value should be no less than the upper bound \(U_r\). Considering these points, the following pricing functions are defined.

4.3.1 Fixed Pricing Function

Fixed pricing methods produce price values that do not differentiate customers’ characteristics. They are not volume dependent, and are not based on real-time market conditions. Currently, most providers follow simple fixed pricing scheme to charge users who are asking for services. Fixed prices are easier to understand and more straightforward for users in the cloud system. However, fixed prices could not be considered as a fair method to all users,
because not all users have the same needs for resources, and not all users can afford the same price \[68\]. Moreover, fixed prices do not permit the provider to give specific incentives via differentiated pricing methods based on distinct user demands \[69\]. In the fixed price method, the seller defines prices for resources that could be prohibitive, it can lead to a reduced customer base and decrease in revenue and profits. In this thesis, we try to provide different constant values for the fixed pricing method, considering the system properties. We define fixed prices for each time slot in order to charge users based on that price. Then, we compare the fixed pricing method to other variable pricing functions which are presented in the following.

4.3.2 Linear Pricing Function

Linear pricing function is popular in economics because it is simple and easy to handle mathematically. They are easy to implement, hence commonly used in practice. In this type of charging functions, the user is charged on the basis of resource usage. In the following, we define a usage-based linear pricing function for charging the users who receive their requested resources in the cloud system.

\[
p_r(a_r(t)) = \frac{U_r}{C_r} a_r(t) + \frac{L_r}{2}
\]

(4.12)

where \(a_r(t)\) is the allocated resource of type-\(r\) at time \(t\). \(C_r\) is the total available capacity of type-\(r\) resource. When \(a_r(t) = 0\), at \(t = 0\), the value of the function is \(\frac{L_r}{2}\) as the lowest value and for \(a_r(t) = C_r\) the value of the function is equal to \(U_r + \frac{L_r}{2}\).

4.3.3 Quadratic Pricing Function

This pricing function is defined as a quadratic function of allocated resource type-\(r\) at time \(t\). This function of \(a_r(t)\) is in the form of \(f(x) = ax^2 + bx + c\), where \(a\), \(b\) and \(c\) are constants.

\[a = \frac{1}{(C_r)^2}\] where \(C_r\) is the total capacity of resource type-\(r\). \(b = \frac{U_r}{C_r}\) and \(c = \frac{L_r}{2}\), where \(U_r = \max_{i \in \{I|c_r^i \geq 0\}} \{b \frac{b}{c_r^i}\}\) and \(L_r = \min_{i \in I} \{\frac{b}{\sum_{r \in R^i} c_r^i}\}\).
A quadratic function is a non-decreasing function for positive values of $a, b$ and $x$, and if $c = 0$ the initial value of the function is 0. Values of $a, b$ and $c$ are positive according to defined function [5.1]. We define our marginal pricing function as a quadratic function with an initial value greater than 0 and smaller than $L_r$ when $a_r(t) = 0, t = 0$. The value of the function increases as the value of $a_r(t)$ increases. When $a_r(t) = C_r$, the value of the pricing function should be at least $U_r$. Thus, the quadratic unit pricing function can be defined as follows:

$$p_r(a_r(t)) = \frac{1}{(C_r)^2} (a_r(t))^2 + U_r \frac{a_r(t)}{C_r} + \frac{L_r}{2} \tag{4.13}$$

It can be seen from the marginal price function that for $a_r(t) = 0, t = 0$ the value of $p_r(a_r(t))$ would be $\frac{L_r}{2}$; while for $a_r(t) = C_r$ the value is $U_r + \frac{L_r}{2}$. Note that the pricing function is independent of user $i$’s bid value and it depends only on the amount of resources that has been sold and user $i$’s demand. Bid-independent pricing function results in a truthful mechanism, discussed in detail in Section 4.4.

### 4.3.4 Logarithmic Pricing Function

This pricing function is inspired by logarithmic functions, $f(x) = \log(x)$. This function has a non-decreasing growth but not in an aggressive manner. Function $g(x) = a \log(c^x) + b$, for given values $c > 0$, $a > 0$ and $b = 0$, is a non-decreasing function with $g(0) = 0$. Therefore, the following payment function is defined in form of $g(x) = a \log(c^x) + b$, where $a = U_r$, $c = (\frac{U_r}{L_r})^{\frac{1}{C_r}}$, $C_r$ is the total capacity of resource type-$r$, and $b = \frac{L_r}{2}$. Similar to the previous pricing function, function [4.14] is independent of user $i$’s bid value.

$$p_r(a_r(t)) = U_r \log\left(\frac{U_r}{L_r} \frac{a_r(t)}{C_r}\right) + \frac{L_r}{2} \tag{4.14}$$

where $p_r(0) = \frac{L_r}{2}$ and $p_r(C_r) = U_r \log\left(\frac{U_r}{L_r}\right) + \frac{L_r}{2}$.

Experimental analysis for this marginal price function is provided in the next chapter.
4.3.5 Exponential Pricing Function

This marginal pricing function is inspired by function $f(x) = x.e^x + b$, where $b \geq 0$ and $c > 1$ are constants defined according to the problem. Figure 4.1 presents function $f(x) = x.c^x + b$ for given $b > 0$ and $c > 1$. For positive values of $x$, the function starts from initial value $b \geq 0$ and continues in a non-decreasing manner. Thus, it can be used as a pricing function with values between a lower and an upper bound. Let $\sigma > 0$ be a parameter such that $\min_{i \in [n]} l_i = \frac{T}{\sigma}$, where $\frac{T}{\sigma}$ is the minimum job length. The marginal pricing function $p_r(t)$ is defined as follows:

$$p_r(a_r(t)) = \frac{L_r}{e.C_r} a_r(t) \left( \frac{U_r.e}{L_r} \right)^{a_r(t)} + \frac{L_r}{e.R.\sigma}$$

(4.15)

where $U_r = \max_{i \in [n], c^r_i > 0} \left\{ \frac{b}{c^r_i} \right\}$, $L_r = \min_{i \in [n]} \left\{ \frac{b l_i}{\sum_{r \in R} c^r_i} \right\}$ and $e \approx 2.7182818...$ is Euler’s number. $C_r$ is the total capacity of resource type-$r$, $R$ is the total number of resources offered by the provider, e.g. $R = 3$, if the cloud provider offers unit resources of RAM, CPU and storage. Similar to previous pricing functions, pricing function [4.15] is independent of user $i$'s bid value.

When $a_r(t) = 0$ for $t = 0$, this results in $p_r(a_r(t)) = \frac{L_r}{e.R.\sigma}$, and for $a_r(t) = C_r$ the price is $p(a_r(t)) = U_r + \frac{L_r}{e.R.\sigma}$. This pricing function has a non-zero initial value. So even when receiving no resources, the user is charged a small amount. This initial price can be considered as a user’s charge for joining to the auction system in real cloud market. Besides, there is another reason for this initial value which is related to the theoretical analysis of this pricing function. In the next section, we provide the theoretical analysis for competitive ratio of the auction mechanism under this pricing function. In that section, we define the Resource Allocation-Price Relationship, which is used for analyzing the competitive ratio of the method. According to the provided definition, the initial value of the pricing function should be greater than zero. Further information is provided in theoretical analysis section.

Simulation studies of these pricing functions are provided in next chapter. We compare
the pricing functions in terms of competitive ratio, user satisfaction and social welfare.

4.4 Theoretical Analysis

4.4.1 Polynomial Time

**Theorem 3.** The computational complexity of Algorithm PD1 is polynomial.

**Proof.** Lines 2 and 3 in PD1 can be executed in linear time since they are initialization of the primal-dual variables and selected cost function. In line 4, upon the arrival of the $i$th user, the algorithm needs to calculate the price of each slot, which takes the number of time slots in the model, $T$ steps (Line 3 in $A_{schedule}$). Line 4 in $A_{schedule}$ takes $O(TR)$ time to schedule the job and check the capacity limit. Lines 5 and 6 can be done in $O(1)$ steps, since they are assignment of values. Thus the running time of this part of $A_{schedule}$ is $O(TR)$. The body of the if condition (lines 8-13) takes $O(RT)$ to update the dual and primal variables and payment computation. Lines 6-10 in PD1 can be done in constant time. Therefore, the total computational complexity of Algorithm PD1 is $O(IRT)$. $I$ is the number of users in the cloud system, $T$ is the total number of time slots available in the system and $R$ is the number of resources offered by the cloud system.  

45
4.4.2 Truthfulness

In practice, users are often selfish, with a natural purpose to maximize their own utilities. They may lie about their true valuations in the hope of a higher utility. The cloud provider instead pursues the highest social welfare possible, to make everyone in the cloud system happy. Therefore, truthfulness is considered as a desirable property in auction mechanism design, which enables the provider to elicit truthful bids.

**Definition 2.** *(Truthfulness in bidding price)* A cloud auction is truthful in bidding price if for any bid $i$, reporting its true valuation is its dominant strategy, and this strategy always maximizes its utility. $\forall b_i \neq v_i, u_i(v_i) \geq u_i(b_i)$.

**Theorem 4.** The online auction algorithm $PD1$ is truthful under price function $p_r(a_r(t)) = \frac{L_r}{eC_r} \cdot a_r(t) \cdot \left(\frac{U_r}{L_r}\right)^{\frac{w_r(t)}{L_r}} + \frac{L_r}{eK_r}$.  

*Proof. (Truthfulness in bidding price)*: Intuitively, if a price a user gets in an auction is independent of the user’s bid value, the auction is truthful. The following theorem is proved in the literature [70].

**Theorem 5.** An auction is truthful if and only if it is bid-independent auction [70].

In our mechanism, upon the arrival of user $i$, if she wins her bid, the payment that user $i$ will be charged depends only on the amount of resources that has been sold and user $i$’s demand. Therefore, her payment is independent of her bidding price. User $i$ cannot improve her utility by lying about her bidding price since her utility equals her valuation minus the payment. Moreover, algorithm $PD1$ always selects the schedule with the maximum utility among all the schedules for user $i$. Thus, truthful bidding guarantees that each user obtains her maximum utility in algorithm $PD1$.

*(Truthfulness in arrival time)*: A user cannot arrive earlier since her arrival time is the first time she is aware of her demand. On the other hand, the marginal price function increases with the amount of allocated resources. If the bidder delays reporting her arrival, the value
of $t$ increases. Since the other users receive their requested amount during this time, the amount of allocated resources grows, resulting in an increase of $a_r(t)$. Recall that $a_r(t)$ is defined as the amount of allocated resources type-$r$ at time $t$. Increasing in the amount of the allocated resources results in a higher payment for the user. Therefore, a user has no motivation to misreport her arrival time.

(Truthfulness in resource occupation time slots and required amount of resource): If the bidder bids a number that is smaller than the required resource amount and a shorter resource occupation time, it will risk failing to cover demand. Hence no bidder has an incentive to do so. On the other hand, the charging payment is equal to the marginal pricing function times the total amount of demanded resources during the requested time slots. Thus, bidding a larger amount of resources and longer time slots will lead to a higher payment for the user, and a decrease in her utility.

Note that for the logarithmic price function and quadratic price function presented in Sections 4.3.4 and 4.3.3 respectively, following the same procedure of proof for Theorem 10, truthfulness property can be proven.

4.4.3 Competitive Ratio and Competitiveness

As mentioned earlier, an optimization problem can be one of either maximization or minimization. Given any legal input $J$, an algorithm $Alg$ computes a feasible solution $Alg[J] \in S(J)$ for an optimization problem $P$, where $S(J)$ is a set of feasible solutions associated with every input $J$. The corresponding cost with this feasible solution can be denoted by $Alg(J) = C(J, Alg[J])$. Therefore, the optimal algorithm $OPT$ can be defined as $OPT(J) = \min_{O \in S(J)} C(J, O)$ for a minimization problem, or $OPT(J) = \max_{O \in S(J)} C(J, O)$ for a maximization problem.

The algorithm $Alg$ is a $c$-competitive algorithm for a minimization problem if there is a constant $\alpha \geq 0$ for which $Alg(J) - c\cdot OPT(J) \leq \alpha$ for all legal inputs. This algorithm
is called an *asymptotic c-approximation* algorithm. For a maximization problem, $\text{Alg}(J)$ represents the profit of $\text{Alg}$. Thus similar to minimization problem, it is required that $\text{OPT}(J) - c\cdot\text{Alg}(J) \leq \alpha$. Note that in both types of problems, the approximation factor $c$ is greater than or equal to 1; the closer the factor is to 1, the better the approximation is \[56\]. $\text{Alg}$ is *strictly c-competitive*, if the additive constant $\alpha$ is equal to zero ($\text{Alg}(J) \leq c\cdot\text{OPT}(J)$).

Note that the constant $\alpha$ becomes insignificant as longer and longer initial subsequences are considered \[56 \ 57\].

A $c$-competitive online algorithm $\text{Alg}$ is considered as a $c$-approximation algorithm with the restriction that $\text{Alg}$ must be computed online. For each input $J$, an algorithm which is $c$-competitive is guaranteed to incur a result within a factor of $c$ of the result of optimal offline algorithm. The competitive ratio is at least 1, and the smaller it is, the better the $\text{Alg}$ performs in comparison to the $\text{OPT}$ algorithm. In the literature, if an $\text{Alg}$ is $c$-competitive, it is said that the $\text{Alg}$ achieves a *competitive ratio* $c$. Therefore, an algorithm is called competitive if it achieves a *constant* competitive ratio $c$. Competitive ratio $c$ can be a function of the parameters of our problem. However, it must be independent of the input $J$ \[56 \ 57\]. An example of a scheduling problem is provided by Borodin et al. \[56\], where for a scheduling problem with $M$ machines, the competitive ratio $c$ depends on $M$. However, it cannot depend on the number or type of jobs being scheduled.

4.4.4 Theoretical Analysis of Competitive Ratio

In this section, we present the theoretical analysis of $c$-competitive ratio for the exponential marginal pricing function defined in previous section. Note that this theoretical analysis holds for the exponential pricing function, and not the other pricing functions.

Exponential Pricing Function:

Recall that $\sigma > 0$ was defined as a parameter such that $\min_{i \in I} l_i = \frac{T}{\sigma}$, where $\frac{T}{\sigma}$ is the minimum job length. $R \geq 1$ is the number of resources available to be allocated to users,
e.g. $R = 3$, if the provider offers resources including RAM, CPU and storage. $C_r$ is the total capacity of resource type-$r$ available at the provider. $e$ is Euler’s number. Thus, $e.R.\sigma > 1$, and the marginal price $p_r(t)$ was proposed as:

$$p_r(a_r(t)) = \frac{L_r}{e C_r} a_r(t) \left( \frac{U_r e}{L_r} \right)^{a_r(t)} + \frac{L_r}{e R \sigma}$$

(4.16)

with $U_r = \max_{i \in [I]: c_i > 0} \{ \frac{b_i}{L_r^c i} \}$ and $L_r = \min_{i \in [I]} \{ \frac{b_i}{e \sum_{r \in R} c_r} \}$

It can be seen from the formulation that $a_r(t) = 0$ results in $p_r(0) = \frac{L_r}{e R \sigma}$ at $t = 0$.

Theoretical Analysis:

The primal-dual analysis framework is discussed in the following lemma, which will be used for the proof of the competitive ratio. We denote the optimal objective values of ILP (4.8) and (4.9) with $OPT_1$ and $OPT_2$ respectively. The result of these two ILP programs is the same, i.e., $OPT_1 = OPT_2$. Let $P_i$ and $D_i$ denote the objective values for primal ILP (4.9) and dual problem (4.10) respectively returned by the algorithm mechanism after processing user $i$’s bids. Let $P_0$ and $D_0$ be the initial values. Then, the final primal and dual objective values from the algorithm can be denoted by $P_I$ and $D_I$, respectively. Note that $I$ represents the total number of users. The following lemma can be proven for a given $P_i$ and $D_i$.

**Lemma 6.** If the offline optimal social welfare $OPT_2 \geq \frac{T}{\sigma} L_r C_r$, $D_0$ computed by the algorithm is at most $\frac{OPT_2}{e}$.

**Proof.** First, the assumption of the lower bound on the offline optimal social welfare should be defined. The assumption means that there are enough jobs to exhaust at least one of the resource types at a time slot. We define $\frac{T}{\sigma}$ as the minimum job length and $L_r$ is defined as the minimum value per unit of type-$r$ resource per unit of time. Therefore, $\frac{T}{\sigma} L_r C_r$ is the minimal social welfare generated by bids that request type-$r$ resources. Initial objective value of the dual problem (4.10) computed by the algorithm is defined as:

$$D_0 = \sum_{r \in [R]} \sum_{t \in [T]} C_r p_r(t) = \sum_{r \in [R]} \sum_{t \in [T]} \frac{C_r L_r}{e R \sigma} = \frac{1}{e} \sum_{r \in [R]} \frac{TC_r L_r}{\sigma R} \leq \frac{1}{e} OPT_2.$$
Lemma 7. The algorithm is $\frac{e}{e-1}c_1$-competitive in terms of social welfare, if there exists a constant $c_1 \geq 1$ such that 1) $P_i - P_{i-1} \geq \frac{1}{c_1}(D_i - D_{i-1}) \forall i$, 2) $D_0 \leq \frac{OPT_2}{e}$ and 3) $P_0 = 0$.

Proof. According to weak duality, we have $D_I \geq OPT_2$. Moreover, by summing up the inequalities for each user $i$, we have $P_I - P_0 = \sum_i (P_i - P_{i-1}) \geq \frac{1}{c_1} \sum_i (D_i - D_{i-1}) = \frac{1}{c_1} (D_I - D_0)$. In addition, according to our assumption $D_0 \leq \frac{OPT_2}{e}$. Therefore, $P_I \geq (1 - \frac{1}{e}) \frac{1}{c_1} OPT_2 = (1 - \frac{1}{e}) \frac{1}{c_1} OPT_1$. Thus, the algorithm is $\frac{e}{e-1}c_1$-competitive. Note that in this formulation, $e$ is Euler’s number, used in the constant part of the pricing function, $L_r e R$.

Later, the value of $c_1$ is determined in theorem 10. Note that $c_1 \geq 1$ because its a factor of competitive ratio and the competitive ratio for online maximization problem results in a value greater than 1. □

As the next step, a Resource Allocation-Price Relationship is defined, and we show that if it holds for a given $c_1$, then the primal and dual objective values achieved by the algorithm is able to satisfy the inequality in Lemma 7.

Definition 3. Considering $c_1 \geq 1$, the Resource Allocation-Price Relationship for the algorithm is $p_{i-1}^r(t)(a_i^r(t) - a_{i-1}^r(t)) \geq \frac{1}{c_1} C_r(p_i^r(t) - p_{i-1}^r(t)), \forall t \in k, \forall i \in [I], \forall r \in [R]$.

Recall that $a_i^r(t)$ is the amount of allocated resource of type-$r$ after accepting user $i$’s task and $p_i^r(t)$ denotes the price of type-$r$ resource after allocating resources to user $i$.

Lemma 8. If the Resource Allocation-Price Relationship holds for a given $c_1$, then the algorithm can guarantee $P_i - P_{i-1} \geq \frac{1}{c_1}(D_i - D_{i-1}), \forall i \in [I]$.

Proof. Assume user $i$’s bid is considered to be accepted or rejected, while let $k$ be the schedule for user $i$’s job. In case of rejection for user $i$, $P_i - P_{i-1} = D_i - D_{i-1} = 0$. If user $i$’s bid is accepted, the increment of primal objective value is $P_i - P_{i-1} = b_{ik}$. Moreover, constraint
(3.10.a) will be tight after accepting user $i$’s bid with schedule $k$ by applying Algorithm 1. Thus, $b_{ik} = u_i + \sum_{r \in R} \sum_{t \in k} (a_r^i(t) - a_r^{i-1}(t))p_r^{i-1}(t)$. The increment of the dual objective value is equal to $D_i - D_{i-1} = u_i + \sum_{t \in k} \sum_{r \in R} C_r(p_r^i(t) - p_r^{i-1}(t))$. Summing up the formula over all $t \in k$ and $r \in R$, results in $P_i - P_{i-1} \geq u_i + \frac{1}{c_1}(D_i - D_{i-1} - u_i)$. $c_1 \geq 1$ and $u_i \geq 0$, thus $P_i - P_{i-1} \geq \frac{1}{c_1}(D_i - D_{i-1})$. 

Next, we find the corresponding $c_{1,r}$ for each resource $r$ since the value of approximation ratio $c_1$ is the maximum value of all $c_{1,r}$, not any of them specifically. We consider the following assumption in order to compute the approximation ratio for each $r$: The amount of requested amount of resources by user $i$ is a small number compared to the total available capacity, $c_r^i << C_r$. Thus, $a_r^i(t) - a_r^{i-1}(t)$ can be expressed as $d_a_r(t)$. Where $d_a_r(t)$ represents an instantaneous rate of change in amount of $a_r(t)$.

**Definition 4.** The differential Resource Allocation-Price Relationship for algorithm with $c_{1,r} \geq 1$ is $p_r(t) d_a_r(t) \geq \frac{c_r}{c_{1,r}} dp_r(t), \forall i, r, t$.

**Lemma 9.** For $c_{1,r} = \ln(\frac{U_r}{L_r}) + 1$ and the above marginal pricing function, the differential Resource Allocation-Price Relationship is satisfied.

**Proof.** The marginal pricing function is $p_r(a_r(t)) = \frac{L_r}{ec_r} a_r(t) (\frac{U_r}{L_r})^{\frac{a_r(t)}{c_r}} + \frac{L_r}{ec_r}$, which is inspired by a function of form $f(x) = axc^x + b$ where $a = \frac{L_r}{ec_r}$, $c = \left(\frac{U_r}{L_r}\right)^{\frac{1}{c_r}}$, and $b = \frac{L_r}{ec_r}$. The derivative of this function is equal to $f'(x) = ac^x(ln(c)x + 1)$. The derivative of our marginal price function is

$$dp_r(t) = p_r'(a_r(t))da_r(t) = \left(\frac{L_r}{ec_r} \frac{U_r}{L_r} \frac{a_r'(t)}{c_r} (a_r(t)ln(\frac{U_r}{L_r})^{\frac{1}{c_r}} + 1)\right)da_r(t) \quad (4.17)$$

The differential Resource Allocation-price Relationship is:
\[
\left( \frac{L_r}{eC_r} a_r(t) \left( \frac{U_r e}{L_r} \right)^{\frac{a_r(t)}{c_r}} + \frac{L_r}{eR\sigma} \right)da_r(t) \\
\geq \frac{C_r}{c_{1,r}} \left( \frac{L_r}{eC_r} \left( \frac{U_r e}{L_r} \right)^{\frac{a_r(t)}{c_r}} \right) \left( a_r(t)ln\left( \frac{U_r e}{L_r} \right)^{\frac{1}{c_r}} + 1 \right)da_r(t) \\
\geq \frac{C_r}{c_{1,r}} \left( \frac{L_r}{eC_r} \left( \frac{U_r e}{L_r} \right)^{\frac{a_r(t)}{c_r}} \right) \left( a_r(t)ln\left( \frac{U_r e}{L_r} \right)^{\frac{1}{c_r}} \right)
\]

\[
c_{1,r} \geq \frac{L_r}{eC_r} a_r(t) \left( \frac{U_r e}{L_r} \right)^{\frac{a_r(t)}{c_r}} + \frac{L_r}{eR\sigma}
\]

factoring \( \frac{L_r}{e} \):

\[
c_{1,r} \geq ln\left( \frac{U_r}{L_r} \right) \frac{\left( \frac{U_r e}{L_r} \right)^{\frac{a_r(t)}{c_r}}}{\frac{U_r e}{L_r} \frac{a_r(t)}{c_r}} + \frac{1}{R\sigma}
\]

Note that the value of \( R \) is an integer, representing the number of types of resources. \( \frac{1}{R\sigma} \) will have a small value so it can be ignored from the formulation, resulting in:

\[
c_{1,r} \geq ln\left( \frac{U_r}{L_r} \right) + 1
\]

\[\text{(4.19)}\]

**Theorem 10.** The online auction Algorithm 1 is \( \frac{e}{e-1} \)-competitive in social welfare with \( c_1 = max_{r \in [R]} \{ ln\left( \frac{U_r}{L_r} \right) \} + 1 \).

**Proof.** By Lemma \[\text{6}\], \( c_1 \) satisfies the Differential Resource Allocation-Price Relationship. Under the assumption that \( da_r(t) = a^i_r(t) - a^{i-1}_r(t) \) is very small compared to the total available capacity of resource type-\( r \) (\( C_r \)), we have \( dp_r(t) = p'_r(a_r(t))da_r(t) = p^i_r(t) - p^{i-1}_r(t) \). Therefore, we can obtain that the Resource Allocation-Price Relationship holds for \( c_1 \). Then, by Lemma \[\text{6}\], Lemma \[\text{7}\] and Lemma \[\text{8}\] the theorem follows. \[\Box\]
Thus the algorithm is $c$-competitive where $c = \frac{e}{e-1}c_1$. In the next chapter, we will examine these pricing functions with simulation studies.
Chapter 5

Empirical Studies

In this chapter, we conduct evaluation of the proposed pricing functions through simulation studies. The results of the simulations are discussed to examine the efficiency of the proposed marginal pricing functions in Chapter 4, and study the performance of the mechanism using each of this pricing methods. In order to demonstrate the efficiency of the method, we implemented the mechanism with different pricing functions and studied it in terms of competitive ratio, user satisfaction, and social welfare. The simulation results demonstrate a better efficiency for the exponential pricing function and illustrate the validity of the theoretical analysis provided in the previous chapter.

5.1 Performance Evaluation for Different Pricing Functions

In order to simulate the auction environment, we use synthesized data which contains the information for each job, including the start time (arrival time), $a_i$, execution duration, $l_i$, user $i$’s deadline, $d_i$, and resource demands (CPU, RAM, storage, etc.), $c_r$. The online algorithm $PD1$ for system model presented in previous chapter is simulated in MATLAB. The offline optimal algorithm for the system model is also simulated. The results of Alg $PD1$ is compared to the corresponding results of optimal offline algorithm to calculate the competitive ratio. The simulation codes for implementing the above two algorithms are provided in Appendix B.

We map each job into a bid, arriving sequentially in 24 hours, $T = 24$. Users’ job deadlines, $d_i, \forall i \in [I]$, are generated uniformly at random between its arrival time and the system end time $T$. We assume that each user’s job consumes $[1, 16]$ slots, $1 \leq l_i \leq 16$, while each time slot is assumed to be 5 minutes. The value of $\sigma = \frac{T}{\min_i \{l_i\}}$ is calculated
according to the formula. $e$ is Euler’s number. The amount of requested resources, $c_i^r$ are selected in the range of $[0,1]$ which is normalized according to the amount of the resources. Thus, the demand for CPU, RAM and storage units is normalized so that the maximum capacity is 1. We adopt the distribution and bid generation technique in [36] to generate combinations of resource request and corresponding bid values. The data in CATS [36] are based on empirical data about auction mechanism systems. The bidding price of each job equals its overall resource demand times unit prices chosen in the range of $[L_r, U_r]$. In this model we assume we have two types of resources, resulting in $R = 2$. The number of users, $I$, are varying in the range of $[40, 100]$, with step 10.

In our simulations, we assume $L_r = 1$ and $U_r = 30$ by default. However, these values will be changed for examining the competitive ratio of each payment method under different ratios of $\frac{U_r}{L_r} = 10$, $\frac{U_r}{L_r} = 30$, $\frac{U_r}{L_r} = 60$. We examine the performance of $PD_1$ in terms of competitive ratio, user satisfaction and social welfare. Note that the results provided in Sections 5.1.1, 5.1.2 and 5.1.3 are the average of results driven from 10 runs of simulations for each method.

5.1.1 Competitive Ratio

In this section we evaluate the algorithm in terms of competitive ratio. Recall that the competitive ratio is defined as the ratio between the online solution and the offline optimum.

1) Fixed-Price Function: Currently, most providers in real world cloud markets follow simple fixed-price schemes to charge users with demands for resources. Recall that fixed pricing methods produce price values that do not differentiate customer characteristics. Moreover, they are not volume dependent, and are not based on real-time market conditions. As mentioned in Table 2.3 in Chapter 2, fixed price methods can be divided in three different pricing models as follow: 1) pay per use; 2) menu price; and 3) subscription. For instance, Amazon uses pay per use fixed pricing [68]. The most prevalent method of
pricing in cloud systems is pay-per-use, which is based on units with constant price. In subscription, users sign a contract based on constant price of service unit and longer period of time. In this section, following pay-per-use pricing model, we define different price values in order to examine the efficiency of the method under fixed-price. Therefore, instead of defining a marginal unit pricing function, we consider constant prices for units of resources. The competitive ratio for the fixed-price function is provided in Fig. 5.1.

![Figure 5.1: Competitive ratio of fixed pricing function.](image)

It can be seen that the fixed-price method of charging users in the cloud system is inefficient in terms of competitive ratio. In this mechanism, we charge users by different values of $U_{\text{max}}$. As the number of users increases, the value of competitive ratio increases as well; and for greater values of $U_{\text{max}}$ the competitive ratio increases.

2) Linear Pricing Function: Multiple providers in real world cloud markets follow usage-based linear pricing methods to charge users for the amount of resource they receive as their usage. In linear pricing methods, the users are charged based on the amount of received resources, hence differs from the flat rate mechanisms. These mechanisms are resource
dependent, but they are not based on real-time market conditions \cite{68}. In this section, we define different price values in order to examine the efficiency of the method under linear pricing method, based on the formula defined in previous chapter, \( p_r(a_r(t)) = \frac{U_c}{c_r} a_r(t) + \frac{L_r}{r} \). The competitive ratio for the linear pricing function is provided in Fig. 5.2, where \( U_c \) is the coefficient of the linear function. Note that the y axis, representing competitive ratio, for this price function is different from the previous graph and the graphs for other pricing functions in the following.

![Figure 5.2: Competitive ratio of linear pricing function.](image)

It can be seen that the linear pricing function for charging users in the cloud system is inefficient in terms of competitive ratio. However, compared to the fixed pricing method, linear pricing method provides lower values of competitive ratio. Thus, the mechanism under linear pricing function achieves better performance than the algorithm under fixed pricing method.

3) Quadratic Pricing Function: The quadratic marginal pricing function was defined
as:

\[ p_r(a_r(t)) = \frac{1}{(C_r)^2} \cdot (a_r(t))^2 + U_r \cdot \frac{a_r(t)}{C_r} \]

where \( U_r = \max_{i \in [I]} \{ \frac{b_i}{c_i} \} \) and \( L_r = \min_{i \in [I]} \{ \sum_{r \in R} \frac{b_i}{c_i} \} \)

Figure 5.3: Competitive ratio of quadratic pricing function for different \( U_r/L_r \).

Recall that the competitive ratio is defined as the ratio between the online solution and the offline optimum. Fig. 5.3 shows the competitive ratio of the algorithm using quadratic marginal pricing function for different values of \( \frac{U_r}{L_r} \). The competitive ratio fluctuates as the number of users increases. The marginal price function is defined based on the real values of \( U_r \) and \( L_r \). As the ratio of \( \frac{U_r}{L_r} \) increases, the value of the competitive ratio is ascending as well but not in a monotone manner. By increasing the number of users, the competitive ratio increases to values greater than 2 when \( \frac{U_r}{L_r} = 10 \). When the ratio of \( \frac{U_r}{L_r} \) is equal to 30 and 60, for almost all number of users, the competitive ratio is greater than 2, and increases with the increase of \( \frac{U_r}{L_r} \). The values of competitive ratio is higher with smaller number of users, when the number of users is 40 or 50, this can happen because of the fact that small number of users can result in the small number of bid options offered from users. So the
provider does not have a great number of bid options to choose among them. Compared to the two discussed pricing functions, the quadratic pricing function results in lower values of competitive ratio, hence the algorithm under this payment achieves better performance than the algorithm under fixed and linear pricing functions. However, the competitive ratios for larger number of users is greater than 2 for this pricing function as well.

4) **Logarithmic Pricing Function**: The logarithmic marginal pricing function was defined as:

\[ p_r(a_r(t)) = U_r \cdot (\log \left( \frac{U_r}{L_r} \right) \cdot \frac{a_r(t)}{c_r} ) + \frac{L_r}{2} \]

where \( U_r = \max_{i \in [I], c_i} \frac{b_i}{c_i} \) and \( L_r = \min_{i \in [I]} \left\{ \frac{b_i}{\sum_{r \in R} c_r} \right\} \)

Fig. 5.4 shows the competitive ratio of the algorithm with logarithmic marginal pricing function for different values of \( \frac{U_r}{L_r} \). The competitive ratio increases with increase in the number of users. It can be seen that as the ratio of \( \frac{U_r}{L_r} \) increases, the value of competitive ratio increases as well. Note that for a smaller number of users, the competitive ratio value remains at a low level < 2. The ratio increases to values larger than 2 with increase in the number of users and increase in \( \frac{U_r}{L_r} \) ratio. The values of competitive ratio is higher with smaller number of users, when the number of users is 40, because of the fact that small number of users can result in few bid options offered from users. So the provider does not have many bid options to choose among them.

Compared to the other discussed pricing function, the logarithmic pricing function results in lower values of competitive ratio, hence the algorithm under this payment achieves better performance than the other discussed pricing functions. However, the competitive ratios for larger number of users is greater than 2 for this pricing function as well.

5) **Pricing Function presented by Zhou et al.**: In this section, we examine the competitive ratio of the pricing function presented by Zhou et al. [3]. This pricing function is provided in the correction section on their work and has not been examined by simulation studies. We use this pricing function to compare with our proposed pricing functions. The
Figure 5.4: Competitive ratio of logarithmic pricing function for different $U_r/L_r$.

pricing function in [1] is as follows:

$$p_r(a_r(t)) = \frac{L_r}{\sigma R e} \left( \frac{e R a U_r}{L_r} \right)^{\frac{\sigma(t)}{c_r}}$$

where $U_r = \max_{i \in [I], c_i} \left\{ \frac{b_i}{c_i} \right\}$ and $L_r = \min_{i \in [I]} \left\{ \frac{b_i}{\sum_{r \in [R]} c_{r_i}} \right\}$. $R$ is a non-negative integer presenting the number of different types of resources available at the cloud provider, i.e. $R = 2$, if the cloud provider offers RAM and CPU to its users. In this simulation we set the value of $R = 2$. $e = 2.71$ is Euler’s number. The value of $\sigma$ is calculated according to $\sigma = \frac{T}{\min_{i \in [I]}}$.

Fig. 5.5 shows the competitive ratio of the algorithm using the above pricing function for different values of $\frac{U_r}{L_r}$. The competitive ratio fluctuates as the number of users increases. In addition, as the ratio of $\frac{U_r}{L_r}$ increases, the competitive ratio increases for almost all of the values. Such a trend agrees with the theoretical bound proven in their work that the value of $\frac{U_r}{L_r}$ determines the competitive ratio. Note that the observed competitive ratio remains at a low level. Compared to the previous pricing functions, this pricing function provide lower values for the competitive ratio.

6) **Exponential Pricing Function** ($f(x) = axc^x + b$): This marginal pricing function
Figure 5.5: Competitive ratio of pricing function presented by Zhou et al. for different $U_r/L_r$. 
was defined as:

\[ p_r(a_r(t)) = \frac{L_r}{e.C_r} a_r(t) \left( \frac{U_r \cdot e}{L_r} \right)^{a_r(t)} + \frac{L_r}{e.R.\sigma} \]

where \( U_r = \max_{i \in [I]} \{ \frac{b_i}{c_i} \} \) and \( L_r = \min_{i \in [I]} \{ \frac{R}{n} \sum_{x \in R} c_r \} \). \( R \) is a non-negative integer presenting the number of different types of resources available at the cloud provider. For this simulation we set the value of \( R = 2 \). \( e \) is Euler’s number and the value of \( \sigma \) is calculated according to \( \sigma = \frac{T}{\min_{i \in [I]} c_i} \). Thus, in our system model, \( \sigma = 24 \).

Fig. 5.6 shows the competitive ratio of the algorithm with the marginal pricing function defined above under different values of \( \frac{U_r}{L_r} \). Note that all the simulation results provided in this chapter as the average of online algorithm simulations for 10 runs. Therefore, these results present the average case of the algorithm, while the theoretical analysis, provided in Chapter 4, present the worst case scenario.

The competitive ratio in Fig. 5.6 fluctuates with an increase in the number of users. Moreover, it can be seen from the results that this pricing function provides lower competitive ratios for average number of users, e.g. 60, 70 users. The observed competitive ratio is much better than the other methods and the theoretical bound. It stays at a low level (< 2). Therefore, it can be concluded that this pricing function provides a near optimal solution, especially for the scenarios with average size of users.

The proposed marginal pricing function is defined based on the values of \( U_r \) and \( L_r \). Thus, we examine the competitive ratio of the algorithm for different values of \( \frac{U_r}{L_r} \). It can be observed that a decrease in the value of \( \frac{U_r}{L_r} \) will result in decreasing values of the competitive ratio, and for increasing \( \frac{U_r}{L_r} \), the competitive ratio fluctuates slightly. Such a trend agrees with the theoretical bound, which is proven in the previous section. Furthermore, over the entire range of values of \( \frac{U_r}{L_r} \) and number of users, this pricing function provides a better competitive ratio in comparison to all previous pricing functions. Therefore, it demonstrates that the proposed pricing function achieves good performance. This observation confirms the analysis in Section 4.4.4 for this marginal pricing function.
Fig. 5.6 demonstrates the value of competitive ratio for all the discussed pricing functions for different number of users when $\frac{U_r}{L_r} = 30$. All the proposed pricing functions provide better competitive ratios compared to the flat-price function and linear pricing method. Thus, they provide efficient solutions, compared to the current prevalent pricing functions, flat-rate and linear pricing. Moreover, it can be seen in Fig. 5.8 that the exponential pricing function provides a better competitive ratio in comparison to all other functions.

5.1.2 User Satisfaction

We next examine user satisfaction for the online algorithm under each pricing function. User satisfaction is defined as the percentage of winning bidders among all the bidders. A greater understanding of user satisfaction can facilitate the design of a customer-friendly resource allocation method, increasing the effectiveness of auction methods; resulting in success of e-commerce applications and businesses [71]. Note that for calculating user satisfaction the percentage of winning bidders is important, and it is not related to the utility. The utility
Figure 5.7: Competitive ratio of all proposed pricing functions for $U_r/L_r = 30$.

Figure 5.8: Competitive ratio of all proposed pricing functions for $U_r/L_r = 30$.
of users can be examined by studying their social welfare, as it is discussed in Section 5.1.3.

User satisfaction of the logarithmic pricing function and the quadratic pricing function are demonstrated in Fig. 5.10 and Fig. 5.9, respectively. When the number of users rises, a decreasing trend is observed in user satisfaction under both pricing functions, because the finite amount of resources can only serve a limited number of users.

In order to achieve a better understanding of the user satisfaction values, we provide multiple statistic information about this graphs as follows. For logarithmic function, the user satisfaction values are in the range of [31%, 63.51%] means that the lowest user satisfaction value for this method, having varying user numbers [40, 100], is 31 and the highest percentage of winners achieved by this method is 63.51%. On the other hand, when \( \frac{U_r}{L_r} = 10 \), user satisfaction mean is 51.44%. For \( \frac{U_r}{L_r} = 30 \) mean value is 48.9% and for \( \frac{U_r}{L_r} = 60 \) mean is 48.6%. It can be seen that as the ratio of \( \frac{U_r}{L_r} \) increases the mean user satisfaction decreases which is according to the trend in Fig. 5.10.

Similar to logarithmic function, user satisfaction values for quadratic pricing function are in range of [43%, 67%] means that the lowest user satisfaction value for this method, having varying user numbers [40, 100], is 43% and the highest percentage of winners achieved by this method is 67%. On the other hand, when \( \frac{U_r}{L_r} = 10 \), user satisfaction mean is 56.14%. For \( \frac{U_r}{L_r} = 30 \) mean value is 54.75% and for \( \frac{U_r}{L_r} = 60 \) mean is 53.1%. It can be seen that as the ratio of \( \frac{U_r}{L_r} \) increases the mean user satisfaction decreases which is according to the trend in Fig. 5.9. Besides, quadratic pricing function provides higher values of user satisfaction compared to the logarithmic function. Similar trend is observed in previous section, where the quadratic function provides lower competitive ratios compared to the logarithmic pricing function.

The same trend can be observed in the user satisfaction of the other two pricing functions, as shown in Fig. 5.11 and Fig. 5.12, respectively. A higher fraction of users are accepted when the number of users is small, because the number of winners is almost fixed due
to the cloud capacity limit. It is observed that the value of $\frac{U_r}{L_r}$ does not affect the user satisfaction percentage for marginal pricing functions; i.e., the winner determination process is not influenced by the change in the value of $\frac{U_r}{L_r}$.

For pricing function proposed by Zhou et.al. [3], the user satisfaction values are in the range of [40%, 69%] means that the lowest user satisfaction value for this method, having varying user numbers [40, 100], is 40 and the highest percentage of winners achieved by this method is 69%. On the other hand, when $\frac{U_r}{L_r} = 10$, user satisfaction mean is 57.4%. For $\frac{U_r}{L_r} = 30$ mean value is 56.08% and for $\frac{U_r}{L_r} = 60$ mean is 56.3%. It can be seen that as the ratio of $\frac{U_r}{L_r}$ increases the mean user satisfaction decreases which is according to the trend in Fig. 5.11. This pricing function provides higher values of user satisfaction compared to the logarithmic and quadratic functions. Similar trend is observed in previous section, where this pricing function [3] provides lower competitive ratios compared to quadratic and logarithmic pricing functions.

User satisfaction values for exponential pricing function are in range of [46.3%, 74.6%] means that the lowest user satisfaction value for this method, having varying user numbers [40, 100], is 46.3% and the highest percentage of winners achieved by this method is 74.6%. On the other hand, when $\frac{U_r}{L_r} = 10$, user satisfaction mean is 60.6%. For $\frac{U_r}{L_r} = 30$ mean value is 59.8% and for $\frac{U_r}{L_r} = 60$ mean is 57.4%. It can be seen that as the ratio of $\frac{U_r}{L_r}$ increases the mean user satisfaction decreases which is according to the trend in Fig. 5.12. Moreover, exponential pricing function provides higher values of user satisfaction compared to all other pricing functions. Similar trend is observed in previous section, where exponential pricing function provides lower competitive ratios compared to others.

5.1.3 Social Welfare

We further examine the social welfare achieved by Algorithm $PD_1$ with different proposed pricing functions. The social welfare is examined for different numbers of users. Fig. 5.13 demonstrates the social welfare for different pricing functions. It can be seen that the online
Figure 5.9: User satisfaction of quadratic pricing function for different $U_r/L_r$.

Figure 5.10: User satisfaction of logarithmic pricing function for different $U_r/L_r$. 

67
Figure 5.11: User satisfaction of pricing function presented in [3] for different $U_r/L_r$.

Figure 5.12: User satisfaction of exponential pricing function for different $U_r/L_r$. 
algorithm $PD_1$ achieves higher social welfare when there are more users participating in the auction. When the number of users grows, the number of bids with larger bidding values will increase as well, leading to a higher social welfare.

![Figure 5.13: Social welfare of the Alg 1. with different number of users and different pricing functions.](image)

The social welfare of the quadratic and logarithmic pricing functions is illustrated in Fig. 5.14 and Fig. 5.15 respectively for different numbers of users and different values for the $U_r/L_r$ ratio. Both the number of users and the value of $U_r/L_r$ influence the social welfare. The bidding price rises with the increase of $U_r/L_r$, since higher value bids result in a higher social welfare.

In order to achieve a better understanding of social welfare of users, we provide multiple statistical information about this graphs as follows. For quadratic function, the social welfare values are in the range of $[542.7, 5145]$. When $\frac{U_r}{L_r} = 10$, the range is $[542\$, 956.2\$] and the mean revenue is 740.12\$. For $\frac{U_r}{L_r} = 30$, the range is $[1195.1, 3223.2]$ and the mean revenue is 2226.86. For $\frac{U_r}{L_r} = 60$, the range is $[2529.3, 5145]$ and the mean revenue is 3929.5. As the ratio of $\frac{U_r}{L_r}$ and the number of users increase, the social welfare increases which is according
to the trend in Fig. 5.14. Note that the range and mean values of social welfare provided by this pricing function is higher than those of logarithmic function. This trend is in accordance to their corresponding competitive ratios.

For logarithmic function, the social welfare values are in the range of $[418.7, 5155]$ (units of revenue, e.g. $\$$). The lowest revenue achieved using this pricing function is 418.7 and the highest revenue is 5155. On the other hand, when $\frac{U_r}{L_r} = 10$, the range is $[418.7, 906.2]$ and the mean of revenue is 729.11. For $\frac{U_r}{L_r} = 30$, the range is $[1482.8, 3011.6]$ and the mean of revenue is 2169.79. For $\frac{U_r}{L_r} = 60$, the range is $[3061.8, 5155]$ and the mean of revenue is 3929.4. As the ratio of $\frac{U_r}{L_r}$ and the number of users increase, the social welfare increases which is according to the trend in Fig. 5.15.

Fig. 5.16 illustrates the social welfare of $PD_1$ when it uses the exponential pricing function for different numbers of users and different values for the $U_r/L_r$ ratio. The same trend as above can be seen for social welfare of exponential pricing functions for different numbers of users and value of $U_r/L_r$. Both the number of users and the value of $U_r/L_r$ influence the social welfare. The bidding price rises with the increase of $U_r/L_r$, since higher value bids result in a higher social welfare. However, the corresponding social welfare values are higher for this pricing function.

For exponential pricing function, the social welfare values are in the range of $[581.7, 5344]$. When $\frac{U_r}{L_r} = 10$, the range is $[581.7, 956.4]$ and the mean revenue is 786.85. For $\frac{U_r}{L_r} = 30$, the range is $[1697.6, 3325.4]$ and the mean revenue is 2388.44. For $\frac{U_r}{L_r} = 60$, the range is $[3561.8, 5344]$ and the mean of revenue is 4326.01. As the ratio of $\frac{U_r}{L_r}$ and the number of users increase, the social welfare increases which is according to the trend in Fig. 5.16. Note that the range and mean values of social welfare provided by this pricing function is higher than those of logarithmic and quadratic functions. This trend is in accordance to their corresponding competitive ratios.

The better performance of the exponential pricing function can be justified as the fact that
Figure 5.14: Social welfare of quadratic pricing function for different $U_r/L_r$.

Figure 5.15: Social welfare of logarithmic pricing function for different $U_r/L_r$. 
the exponential function has an aggressive manner for smaller values of inputs, compared to the logarithmic function and quadratic function. Thus, the resource allocation at the beginning time slots would be more than other functions. More resource allocation results in more user satisfaction, an increase in number of winners and more revenue for the system. Therefore, this mechanism results in a performance closer to optimal mechanism.

In this chapter, we conduct evaluation of the proposed pricing functions through simulation studies. First we compare the pricing functions in terms of competitive ratio. The highest values of competitive ratios are provided by the fixed pricing function and the lowest by the proposed exponential pricing function. Thus, the exponential pricing function provides a better solution compared to the other pricing functions. Moreover, the quadratic pricing function provides better competitive ratio compared to logarithmic pricing function and both of these function provide better performance compared to the linear pricing function and fixed-price method. User satisfaction and social welfare values are in accordance to the result of competitive ratio. The exponential pricing function provides better user satisfaction and higher social welfare compared to other pricing functions. Logarithmic and
quadratic pricing functions are not considered as efficient mechanisms compared to exponential pricing function. However, these two methods provided higher user satisfaction and revenue compared to linear and fixed pricing methods.
Chapter 6

Conclusion and Future Work

6.1 Conclusion and Thesis Summary

Cloud computing is emerging as a new paradigm for providing rapid on-demand access to resources for cloud users. In this thesis, we studied the cloud market, providing resources to users who have computing jobs with completion deadlines. A cloud user bids for future cloud resources to execute her job within the corresponding deadline. In these systems, the provider’s goal is to maximize its revenue or the social welfare using carefully designed allocation and pricing methods. The users’ purpose is to achieve the highest amount of utility with a reasonable price. Therefore, pricing methodologies for charging cloud users play a central role in designing these systems. We designed online auction mechanisms for cloud jobs that execute in an online fashion. We adopt an online scheduling algorithm for resource allocation to cloud users, and design three different marginal pricing functions for charging users in the system. Then, we showed that the pricing functions together with the allocation rule, runs in polynomial time and provides truthfulness guarantee. Competitive ratio analysis was provided for the algorithm under one of the proposed pricing functions, showing that the mechanism can achieve near-optimal social welfare, using this payment method. We then perform simulation studies to evaluate the performance of each method.

In Chapter 1, we explained the motivation of our research and the importance of pricing methodologies in online auctions and resource allocation in cloud computing systems. Further, we mentioned our contributions in this research and explained our thesis plan.

In Chapter 2, we reviewed the necessary background on auction theory and mechanism design. We then provided information on linear programming based algorithms in order to build the foundation for our problem formulation and solution algorithm design. The
resource allocation in cloud computing is defined and discussed in this chapter, while current pricing models for cloud systems were reviewed as well. At the end of this chapter, we covered the simulation methods and data generation for auction mechanisms.

In Chapter 3, the existing literature in the area of resource scheduling, cloud computing pricing methods and online auction mechanisms was reviewed, and the differences between offline and online mechanisms are discussed.

We demonstrated our problem formulation and system model in Chapter 4. This chapter has three main parts. First, there is the mathematical system model, in which we formulate the maximization of social welfare considering system constraints. Then, we provide the bidding language of the model and the online algorithm to be applied to the problem formulation. The proposed pricing functions, which are designed according to system properties, are covered in the next part. In the last part of this chapter, theoretical analysis of the presented methods are provided.

Finally, Chapter 5 demonstrates the simulation results of the auction mechanism under each pricing function with different assumptions. The simulation results are presented and the comparison of the mechanisms under each price function are discussed.

Multiple pricing functions are presented in this thesis. These functions charge users based on different criteria and properties. It can be concluded from this thesis that among different types of pricing functions, fixed pricing functions provide inefficient mechanism for charging users. The output of the system under fixed-price method, is not near optimal in terms of revenue of the system. Another pricing function is linear function based on users’ usages. This function provides more efficient mechanism compared to the fixed-price mechanism. However, it does not provide a good performance compare to other mechanisms. Other pricing mechanisms are quadratic, logarithmic and exponential pricing functions in terms of the amount of resource type−r which is allocated during the mechanism. The efficiency of the mechanism are in ascending order for logarithmic, quadratic and exponential mechanism
respectively. The most efficient mechanism among all of the pricing functions evaluated is the exponential pricing mechanism. This mechanism provides a near optimal solution for the cloud system. Moreover, it can be concluded that there are other factors than pricing functions that influence the efficiency of the system, including number of users in the system, offered bidding values and their maximum and minimum values, and available capacity of the system. The proposed pricing functions can be applied to different auction mechanisms.

6.2 Future Work

Some possible future directions for this research are suggested in the following.

- One may apply the proposed payment functions to other auction design problems where a deadline is involved, and users are charged based on user’s demand and the amount of resource that has been sold such as demand response problems in smart grid.

- The proposed method can also be applied to the online scheduling problems with speed scaling, where each machine runs at different speeds according to energy cost. In this system, it is important how fast the computational resources assigned to the job are, thus the scheduling algorithm together with the pricing method can be applied to these systems.

- The method and result of this work can be used for data-processing job models (e.g, COSMOS, Hadoop), where the system can benefit from uneven and time-varying allocation of resources and payments.

- This model can be extended to include other realistic constraints in cloud market, such as considering different servers available at the cloud provider.
Appendix A

Deriving The Dual Program

The LP we start with is the primal. The dual can be derived from the primal via a mechanical procedure according to [1].

The primal program of our model is as follows:

Maximize \[ \sum_{i \in I} \sum_{k \in \Lambda_i} b_{ik}X_{ik} \quad (A.1) \]

Subject to:

\[ \sum_{t \in T} \sum_{i \in I} c_{i}^r X_{ik} \leq C_r, \forall r \in [R], \forall t \in [T] \quad (A.1a) \]

\[ X_{ik} \in \{0, 1\}, \forall i \in [I], \forall k \in \Lambda_i \quad (A.1b) \]

In order to construct the dual of the primal:

1) First, we need to put the primal in a standard form. So we rewrite the objective as a minimization. Note that a solution that maximizes an objective value also minimizes the negative of that objective.

Minimize \[ -\sum_{i \in I} \sum_{k \in \Lambda_i} b_{ik}X_{ik} \]

2) We rearrange each constraint so that the right-hand side is 0.

Subject to:

\[ \sum_{t \in T} \sum_{i \in I} c_{i}^r X_{ik} - C_r \leq 0, \forall r \in [R], \forall t \in [T] \]

\[ X_{ik} - 1 \leq 0, \forall i \in [I], \forall k \in \Lambda_i \]

3) We need non-negative dual variables for each inequality constraints. We define \( p_r(t) \) and \( u_i \) for the first and second constraints, respectively.
4) Eliminate each constraint and add the term (dual variable)·(left-hand side of constraint) to the objective. Then maximize the result over the dual variables.

\[
\text{Maximize} \quad \text{Minimize} \quad - \sum_{i \in I} \sum_{k \in \Lambda_i} b_{ik} X_{ik} + p_r(t) \left( \sum_{i \in I} \sum_{k \in \Lambda_i} c_i^r X_{ik} - C_r \right), \forall r \in [R], \forall t \in [T] \\
+ u_i (X_{ik} - 1), \forall i \in [I], \forall k \in \Lambda_i
\]

5) The next step is to rewrite the objective so that it consists of several terms of the form (primal variable)·(expression with dual variables), with the remaining terms involving only dual variables, the result is:

\[
\text{Maximize} \quad \text{Minimize} \quad - \sum_{i \in I} u_i - \sum_{r \in R} \sum_{t \in T} C_r p_r(t) \\
+ u_i + \sum_{i \in I} \sum_{k \in \Lambda_i} c_i^r p_r(t) - b_{ik}, \forall i \in [I], \forall k \in \Lambda_i
\]

6) Then, it is needed to remove each term of the form (primal variable)·(expression with dual variables) and replace it with a constraint of the form expression ≥ 0, if the primal variable is non-negative.

\[
\text{Maximize} \quad - \sum_{i \in [I]} u_i - \sum_{r \in [R]} \sum_{t \in T} C_r p_r(t) \\
\text{Subject to:}
\]

\[
u_i \geq b_{ik} - \sum_{t \in k} \sum_{r \in [R]} c_i^r p_r(t), \forall i \in [I], \forall k \in \Lambda_i
\]

\[
p_r(t), u_i \geq 0, \forall i \in [I], \forall t \in [T], \forall r \in [R].
\]

7) At last:

\[
\text{Minimize} \quad \sum_{i \in [I]} u_i + \sum_{r \in [R]} \sum_{t \in [T]} C_r p_r(t) \quad (A.2)
\]
Subject to:

\[ u_i \geq b_{ik} - \sum_{t \in k} \sum_{r \in [R]} c^r_i p_r(t), \forall i \in [I], \forall k \in \Lambda_i \]  \hspace{1cm} (A.2a)

\[ p_r(t), u_i \geq 0, \forall i \in [I], \forall t \in [T], \forall r \in [R]. \]  \hspace{1cm} (A.2b)
MATLAB code for implementing the algorithm is provided in this Appendix. First, implementation code of online algorithm $PD_1$ is provided in first section that implements algorithms $PD_1$ and $A_{schedule}$ for different number of users with multiple bid offers. In order to determine the competitive ratio, the online algorithm should be compared to an offline optimal algorithm. The implementation code for offline optimal algorithm is provided in Section [B.2]. All of the constraints of optimization problem (4.9), are considered in the implementation of offline optimal algorithm.

B.1 Online Auction Algorithm

%%%Online Algorithm PD1%%%

I= 40:10:100; %number of users
T=24; %number of slots
u1=zeros(1,J); %each user’s unit price for type1
u2=zeros(1,J); %each user’s unit price for type2
x=zeros(1,I); %xor bidding rule
z=zeros(2,T); %used resource
p=zeros(2,T); %price

obj=0; %objective value of the function
Umax=10; %maximum value of resource per unit of time t, (U_r)
Umin= 1; %minimum value of resource per unit of time t, (L_r)
LC = 1; % temporary value for corresponding Umin
UC = Umax; % temporary value for corresponding Umax
J = 1; % number of bids per user
f = zeros(J, I); % matrix for bidding price
R = zeros(2, I); % request amount
WW = zeros(1, I); % request time
deadline = zeros(J, I); % deadline value for each user

%% checking the constraints related to deadline and scheduling
for i = 1: I

% deadline for user i

deadline_array = zeros(J, 1);
bid = zeros(J, 1); % bidding price per user
r1 = 1 * rand(1); % random value for resource type 1
r2 = 1 * rand(1); % random value for resource type 2

R(1, i) = r1; % requested amount of resource type 1 by user i
R(2, i) = r2; % requested amount of resource type 1 by user i
w = randi([1, 16], 1); % # of time slots requested by users
WW(i) = w; % # of time slots requested by user i, l_i

% deadline of users

deadline_array(1) = i + w + rand([0, T-w-I], 1);
u1(1) = Umin + (Umax - Umin) * rand(1); % each user's unit price for type1
u2(1) = Umin + (Umax - Umin) * rand(1); % each user's unit price for type1
bid(1)=(u1(1)*r1+u2(1)*r2)*w;               %bid value for each user

%if the user has more than one bid
if J>1
for j=2:J

%deadline of user
    deadline-array(j)=deadline-array(j-1)
    +randi([0, T-dead(j-1)],1);

%unit price for type1
    u1(j)=Umin+(u1(j-1)-Umin)*rand(1);

%unit price for type1
    u2(j)=Umin+(u2(j-1)-Umin)*rand(1);

%bid value for user
    bid(j)=(u1(j)*r1+u2(j)*r2)*w;
end
end

deadline(:,i)=deadline-array;
f(:,i)=bid;
c=zeros(1,T); %cost of resource for each time slot
for t=1:T;
c(t)=p(1,t)*r1+p(2,t)*r2; %price at each slot
\[
c(t) = (p(1, t) + p(2, t)) \quad \text{fixed-price version}
\]

end

l = zeros(J, w);
pay = zeros(1, J); % payment
u = zeros(1, J); % utility

% find schedule for each bid
for j = 1:J
    tau = [];
    cnew = [];

    % find feasible slots between arrival time and deadline
    for t = i:deadline-array(j)
        if z(1, t) + r1 <= 1 && z(2, t) + r2 <= 1
            tau = [tau t];
            cnew = [cnew c(t)];
        end
    end
end

cnew = sort(cnew);
if length(cnew) >= w
    c_{-1} = zeros(1, w);
schedule = [];
% scheduling the requested job according to the time slots
for \(ww=1:w\)
    
    \[\text{index} = \text{find}(c == \text{cnew}(ww));\]
    
    \(nn=1;\)
    
    while(\(\text{ismember}(\text{index}(nn), \text{schedule}) \text{||~ismember}(\text{index}(nn), \text{tau})\))
        
        \(nn=nn+1;\)
        
    end
    
    \(c_{-}l=c(\text{index}(nn));\)
    
    \(\text{schedule}=[\text{schedule} \ \text{index}(nn)];\)
    
end

\(l(j,:)=\text{schedule}; \quad \% \text{set the schedule}\)

\(\text{pay}(j)=\text{sum}(c_{-}l); \quad \% \text{set user payment}\)

\(u(j)=\text{bid}(j)-\text{pay}(j); \quad \% \text{utility calculation}\)
else

\(u(j)=0;\)

end

end

\%\text{choosing the maximum utility for the user and set user as winner}\n
\(\text{win}=\text{find}(u==\text{max}(u),1);\)

\%\text{user will be winner with her maximum utility}\n
if u(win)>0

\(x(i)=1;\)

\(\text{for } ww=1:w\)

\(tt=l(\text{win},ww);\)

\%\text{dual variables Default equal to a_r(t) in this thesis}
\[
\begin{align*}
z(1,tt) &= z(1,tt) + r1; \quad \text{% for resource type 1} \\
z(2,tt) &= z(2,tt); \quad \text{% for resource type 1} \\
%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\% \text{Pricing Function Presented by Zhou et al}\%

\% \text{P(p1,tt)= (Lc/((T/ww)* 2.71 * 5 ))*}
\text{(((2.71*5 * (T/ww) * Uc)/Lc).^(z(1,tt)/(R(1,i))))};

\% \text{P(p2,tt)= (Lc/((T/ww)* 2.71 * 5 ))*}
\text{(((2.71*5 * (T/ww) * Uc)/Lc).^(z(2,tt)/(R(2,i))))};

\text{end}
\text{obj=obj+bid(win);} \% \text{objective function}
\text{end}
\text{end}
\text{obj}\text{sum(x)/I}

\text{B.2 Offline Optimal Algorithm}

\% \text{Offline Optimal Algorithm}\%
nf=reshape(f,1,[1]);
\text{ff=-[ff zeros(1,1*T)];}
deadline=reshape(deadline,1,[]); \% \text{deadline setting}
\text{sizeofd=length(deadline)};
\text{sizeoff=length(ff)};
A1=zeros(I*T,sizeoff);

b1=zeros(I*T,1);

for i=1:I
  for t=1:T
    A1((i-1)*T+t,(i-1)*J+1:(i-1)*J+J)=-deadline((i-1)*J+1:(i-1)*J+J);
    if t>=i
      A1((i-1)*T+t,sizeofd+(i-1)*T+t)=t;
    end
  end
end

b2=zeros(I,1);

A2=zeros(I,sizeoff);

for i=1:I
  A2(i,(i-1)*J+1:(i-1)*J+J)=WW(i);
  A2(i,sizeofd+(i-1)*T+i:sizeofd+(i-1)*T+T)=-1;
end

b3=ones(2*T,1);

A3=zeros(2*T,sizeoff);

for k=1:2
  for t=1:T
    for i=1:I
A3((k-1)*T+t, sizeofd+(i-1)*T+t) = R(k,i);

end

%%%constraint (4.9d)%%%

b4 = ones(I,1);
A4 = zeros(I, sizeoff);
for i=1:I
    A4(i, (i-1)*J+1: (i-1)*J+J) = 1;
end

%%%constraint (4.9d)%%%

intcon = zeros(1, sizeoff);
for j = 1:sizeoff
    intcon(1,j) = j;
end

Aeq = [];
beq = [];
lb = zeros(1, sizeoff);
ub = ones(1, sizeoff);
A = cat(1, A1, A2, A3, A4);
b = cat(1, b1, b2, b3, b4);
[xx, fval] = intlinprog(ff, intcon, A, b, Aeq, beq, lb, ub);
-fval
-fval/obj
Bibliography


