A Price-Incentive Resource Auction Mechanism Balancing the Interests Between Users and Cloud Service Provider

Songyuan Li, Jiwei Huang, *Member, IEEE*, and Bo Cheng, *Member, IEEE*

***Abstract*—For a cloud service provider, it necessitates an emerging cloud ecosystem to consolidate the existing users and attract more potential users, further gaining its market share. Therefore, in this article, we design a price-incentive resource auction mechanism in cloud environment. In response to the cloud resource price, each user synthesizes her bidding budget and QoS requirement, and purchases cloud resources according to her resource demand in a strategic manner. The cloud service provider, meanwhile, can regulate the resource demands of users through conducting a market-based pricing strategy, against too low prices to cover the operational costs (i.e., energy costs) or too high prices resulting in user churn. In virtue of an elaborate market-based pricing strategy, the interests of users and the cloud service provider are balanced. Our price-incentive resource auction mechanism targets to stimulate maximum users willing to purchase resources and perform their applications at the cloud, on the premise of a minimum profit rate guaranteed for the cloud service provider. It is also able to provide budge balance and truthfulness guarantee, and satisfy the envy-freeness. In order to carry out the above objectives, we carefully design the user utility function reflecting the complicated user interest, and formulate our resource pricing and auction problem as a bin packing problem, which has non-polynomial computational complexity. Regarding the NP-hardness of optimization problem and the concavity of user utility, we present a computational-efficient $(1+\epsilon)$-approximate algorithm namely PIRA. Finally, we conduct simulations based on the real-world dataset to validate the effectiveness of our proposed approach.**

***Index Terms*—Cloud computing, resource management, market-based pricing, auction mechanism.**

Manuscript received May 7, 2020; revised September 9, 2020; accepted November 6, 2020. Date of publication November 10, 2020; date of current version June 10, 2021. This work was supported by National Natural Science Foundation of China (No. 61972414), Beijing Nova Program of Science and Technology (No. Z201100006820082), Beijing Natural Science Foundation (No. 4202066), National Key Research and Development Program of China (No. 2018YFB1003800), and Fundamental Research Funds for Central Universities (No. 2462018YJRC040). The associate editor coordinating the review of this article and approving it for publication was N. Kamiyama. *(Corresponding author: Jiwei Huang.)*

Songyuan Li and Bo Cheng are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lisy@bupt.edu.cn; chengbo@bupt.edu.cn).

Jiwei Huang is with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China (e-mail: huangjw@cup.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNSM.2020.3036989, provided by the authors.

Digital Object Identifier 10.1109/TNSM.2020.3036989

I. Introduction

With the increasing prevalence of cloud computing, a variety of cloud service providers (e.g., Amazon EC2 [1], Google Cloud Platform [2], Microsoft Azure [3]) are presented to users who intend to migrate their applications into cloud. Thanks to the technological advance of cloud computing, mainstream cloud service providers usually provide similar services, and are capable of meeting the general service requirements of most cloud users. In this case, a cloud user may have several candidate cloud service providers for choice. In the face of market competitiveness for similar services, it is necessary for a cloud service provider to develop a burgeoning cloud ecosystem, with the aim of maintaining the existing users and alluring more potential cloud users, further increasing its market share. A significant market share can support a cloud service provider's profitability and sustainability, which is conducive to holding a competitive advantage in the cloud marketplace.

Market-based cloud pricing strategy plays a prominent role in regulating the cloud service provider's resource supply and the resource demands of users [4], which can be an engine of constructing an advanced cloud ecosystem. Resource demands of users are usually price sensitive, varied with different cloud pricing settings. When the cloud resource price is set as a relatively lower price, more users are attracted to purchase cloud resources. However, when the cloud service provider raises the cloud resource price, the resource demands of users are then tightened. The cloud service provider's operational costs mainly arise from the energy costs of the cloud infrastructure [5], [6]. With the cloud service provider's operational costs (i.e., energy costs) and resource demands of users fluctuating with time, the cloud service provider would dynamically modulate the resource price, and deliver a competitive resource price [7]. The competitive resource price can not only recoup the cloud service provider's operational costs from users, but also incentivize the maximum users to purchase cloud resources on the cloud platform. With the maximum cloud users attracted, the cloud service provider captures a significant market share in the marketplace.

In this article, we explore the efficacy of market-based cloud pricing strategy by putting forward a price-incentive resource auction mechanism balancing the interests of users and the cloud service provider. Multiple users simultaneously make service requests, and place their bids and QoS requirements

to the cloud. On the one side, each user would pay for the allocated cloud resources, only when her QoS requirement is satisfied by the cloud service provider. On the other side, the cloud service provider receives the service bids from users, and then makes a competitive pricing decision on cloud resources to call on maximum users served at cloud. The user is incentivized to outsource her application at cloud, due to her gained maximum net-utility relating to her bidding budget and QoS requirement. The energy cost incurred by operating the cloud infrastructure is also adopted as an important consideration influencing the pricing setting. The resource price and chargement should overrun the cost, and ensure a particular profit rate which is specified by the cloud service provider. The service bid whose budget is below the cost would not be accepted by the cloud service provider. To sum up, the objective of our resource auction mechanism is to motivate maximum users to host their applications at cloud while a profitability objective requires to be at least guranteed. This way, our cloud ecosystem reaches a balance between the interests of users and the cloud service provider.

In order to develop a robust resource auction mechanism, it is essential to satisfy the properties of budget balance, truthfulness and envy-freeness. Budget balance is usually treated as a primary requirement when designing the resource auction mechanism [8]. It claims that the bidding budget of a user is always sufficient to cover her service payment; otherwise, the proposed auction mechanism is invalid. Besides, the cloud service provider also needs to notice the fraudulent behavior of users. One typical fraudulent behavior may be misreporting the bidding budget with the attempt to be allocated more cloud resources, further gaining a higher QoS. Hence, the resource auction mechanism should provide the truthfulness guarantee, which removes the opportunistic incentive to misreport the bidding budget [9]. In terms of the envy-freeness, it is an inherent fairness criterion in economics. Regarding a cloud pricing setting, users can be allocated the amount of resources which maximize their utilities [10]. In our problem, a net-utility function is elaborately designed for the user, which collectively evaluates the user's QoS gain and economic cost of resource purchase. Therefore, with the guarantee of budget balance, truthfulness and envy-freeness, our price-incentive resource auction mechanism can enhance the user satisfaction and instruct users to truthfully reveal their service bids, further consolidating more cloud users.

However, designing such a resource auction mechanism which achieves the above objective and properties is challenging in three folds. First, it is awkward to satisfy the properties of both truthfulness and envy-freeness in the auction where the cloud service provider simultaneously sell resources to multiple users. As exemplified by the Vickrey Clarke Groves (VCG) auction [11], it provides the truthfulness guarantee, but discriminates between bidders (i.e., users) by selling identical goods (i.e., cloud resources) at different prices, which dissatisfies the requirement of envy-freeness. Second, resource pricing and procurement are coupled. The resource pricing result determines the resource purchase demands of users, and in return, the stimulated resource purchase demands also reflect the market competitiveness of the resource pricing result. Hence, it is required to comprehensively investigate the interplay between resource pricing and procurement. Third, the final QoS gain of a service request is by no means proportional to the amount of allocated resources, usually with concavity [12] instead. As a result, our optimization problem is non-convex, which is computationally intractable to solve out the optimal resource price. Thus, we expect to design a computational-efficient approximate algorithm with a good approximation ratio to attack the intractablity of the optimization problem.

Main contributions of our work are summarized as follows.

- We investigate the interaction between users and the cloud service provider in the process of resource pricing and auction, aiming at stimulating maximum users served at cloud and ensuring the cloud service provider's profitability goal. Based on this, we devise a cloud ecosystem where the main steps in the price-incentive resource auction mechanism are identified.
- To calculate the optimal resource price, we carefully define the net-utility of user, and formulate our resource pricing and auction problem as a non-convex optimization model, where both the NP-hardness and computational intractability incurred by concave utility functions are analyzed.
- We design an approximate algorithm based on the decomposition approach namely `PIRA` to achieve the objective of incentivizing the maximum users served at cloud, on the premise of the cloud service provider's minimum profit rate guaranteed. Having the maximum users served at cloud, the cloud service provider's revenue is also maximized with an $(1+\epsilon)$ approximation ratio.
- We theoretically prove that our mechanism meets the properties of budge balance, truthfulness and envy-freeness, while the computational complexity of `PIRA` algorithm is analyzed. We also conduct experimental validation based on the real-world dataset to evaluate the performance of our proposed approach.

The rest of article is organized as follows. We first discuss related works in Section II. Then, in Section III, we introduce our system model and formulate our optimization problem. In Section IV, we put forward our price-incentive resource mechanism and theoretically prove its properties. In Section V, we extensively conduct simulating experiments based on the public real-world dataset to verify the effectiveness of our resource auction mechanism. Finally, we state the concluding remark and look ahead our future work in Section VI.

II. Related Work

A. Market-Based Cloud Pricing

Market-based pricing strategy has been launched into real-world cloud enterprise. One of the prominent is the Amazon EC2 Spot Instance [13], which is aimed to take advantage of the idle EC2 instances. The Amazon EC2 operator gradually modulates the price of Spot Instances on the basis of long-term trends in supply and demand for Spot Instance capacity, which is normally lower than the price of on-demand instances. To make idle EC2 instances utilized, Yang *et al.* [14] designed a

reserved-instance reselling algorithm for the cloud user, where the reserved instance which was highly likely to be unutilized in the future would be resold as the Spot Instance to reduce the waste of unused reservations. Khodak *et al.* [15] exploited the price advantage of Spot Instance to save the economic cost of cloud user, where the user purchased resources based on the predicted composite demand of Spot and on-demand instances making applications executed within the acceptable QoS level. Kamiyama [16] got enlightened by the mechanism of Amazon EC2 Spot Instances, and then proposed a novel VM trading approach amongst multiple cloud providers which allowed idle VM instances transferred to other providers for sufficient use. To sum up, the recent study on the Amazon EC2 Spot Instance mostly focuses on the resale of idle resources, or takes the price advantage of Spot Instance to bring down the operational cost of cloud users.

Moreover, there are also several literatures concerning the design of market-based pricing strategy itself. Compared with the static pricing strategy, the market-based pricing strategy can capture and regulate the fluctuation of supply and demand for cloud resources, thereby more effectively maximizing the revenue/profit earned by the cloud service provider. Wan *et al.* [17] designed a reactive pricing strategy for cloud resources with the aim of profit maximization, which dynamically modulated the cloud server price in response to the fluctuation of energy costs and user demands. Cong *et al.* [18] explored the users' SLA requirements as user-perceived values, and then presented a market-based server pricing model dynamically optimizing the cloud service provider's profit. Zheng *et al.* [19] proposed a dynamic bandwidth pricing strategy for revenue maximization in inter-datacenter networks, where the bandwidth resource was time-independently priced based on varied bandwidth demands of users. Wang *et al.* [20] combined the market-based pricing strategy with effective VM capacity modulation to consolidate the cloud profitability from the latency-tolerant users.

Besides, market-based pricing strategy is also functioned in timely recouping fluctuant energy costs from cloud users. The data center operated by the cloud service provider is normally situated in a deregulated electricity marketplace, where the electricity price is evolving stochastically over time [21], [22]. Aldossary *et al.* [23] surveyed several energy-aware pricing schemes, and identified the importance of designing a cloud pricing strategy which should accurately reflect the fluctuant electricity price. Nasiriani *et al.* [24] learned the fluctuant user workload causing the peak of energy cost, and thus developed a peak pricing scheme which fairly distinguished each cloud user's contribution to the peak energy cost. Sarker *et al.* [25] paid regard to unpredictable electricity load and congestion in the urban electric vehicle environment, and thus developed a nonlinear power pricing strategy to regulate the electric vehicle's power request with the social welfare maximized. Qiu *et al.* [26] aimed to achieve green cloud computing, and thus designed a discriminatory pricing policy for cloud resources which both maximized the profit earning and minimized the instant energy costs, with user demands also taken good care of.

B. Resource Auction Mechanism

Auction is a powerful tool to characterize the market behaviors of cloud resource pricing and allocation. A good resource auction mechanism can precisely model the interaction between resource providers and buyers, as well as perfectly match the resource supply and demand with an effective pricing standard regulated. Huang *et al.* [27] investigated the competition between network and cloud providers from the perspective of an non-cooperative game, and gained an economic understanding towards cloud auction. Landa *et al.* [28] designed a Vickrey-auction-based service provisioning scheme, aimed to handle the resource competition across users in distributed clouds. Jiang *et al.* [29] studied the dynamic characteristics of IaaS consumers and providers from the Google Cluster Trace, and then proposed a stochastic resource pricing and allocation mechanism based on the VCG auction. Sun *et al.* [30] formulated the resource sharing mechanism amongst cloudlets as a sealed-bid bilateral auction, where each resource transaction was recorded in contract. Lu *et al.* [31] paid concern on the two-sided cloud market with multiple alternative cloud providers, based on which a double auction was designed to match the requirements from both users and providers. Zhang *et al.* [32] proposed a profit-driven two-stage auction process with regard to tiered cloud storage, where the operations of data storage and access were both taken into account. Hosseinalipour and Dai [33] modelled the market-oriented interactions amongst cloud users, managers and providers with a two-stage auction model, where the interaction between cloud users and managers was characterized by the options-based sequential auction.

Furthermore, several essential auction properties, including budget balance, truthfulness and envy-freeness, should be taken into account when designing the resource auction mechanism. *Budget balance* implies that the bidding budget of user would not be violated, which is generally seen as a primary requirement of resource auction mechanism. Zheng *et al.* [8] leveraged the idea of proportional sharing to admit the budget balance, where the overall bidding budget from users was sufficient to cover the resource instances' procurement cost. Arabnejad *et al.* [34] studied the budget-constrained workflow scheduling in clouds, where the total workflow budget was distributed across various partitioned task subsets. *Truthfulness* is a critical property used for preventing market manipulation. Feng *et al.* [35] presented an effective learning approach for measuring incentive compatibility (i.e., truthfulness) in auctions, which provided great technical guidance to the auction mechanism validation. Zhang *et al.* [36] exploited the truthfulness of VCG auction into the computation offloading problem on mobile cloud computing, where the Lyapunov optimization technique was applied to attack the high complexity of VCG mechanism. Zhu *et al.* [37] proposed an online auction mechanism for underutilized IaaS instances for the sake of maximizing cloud resource utilization, in which the truthfulness was ensured by sequential posted price mechanisms. *Envy-freeness* indicates such a fair resource allocation scheme that each user prefers the resource allocation of her own to that of others. Baranwal and Vidyarth [38] equalised

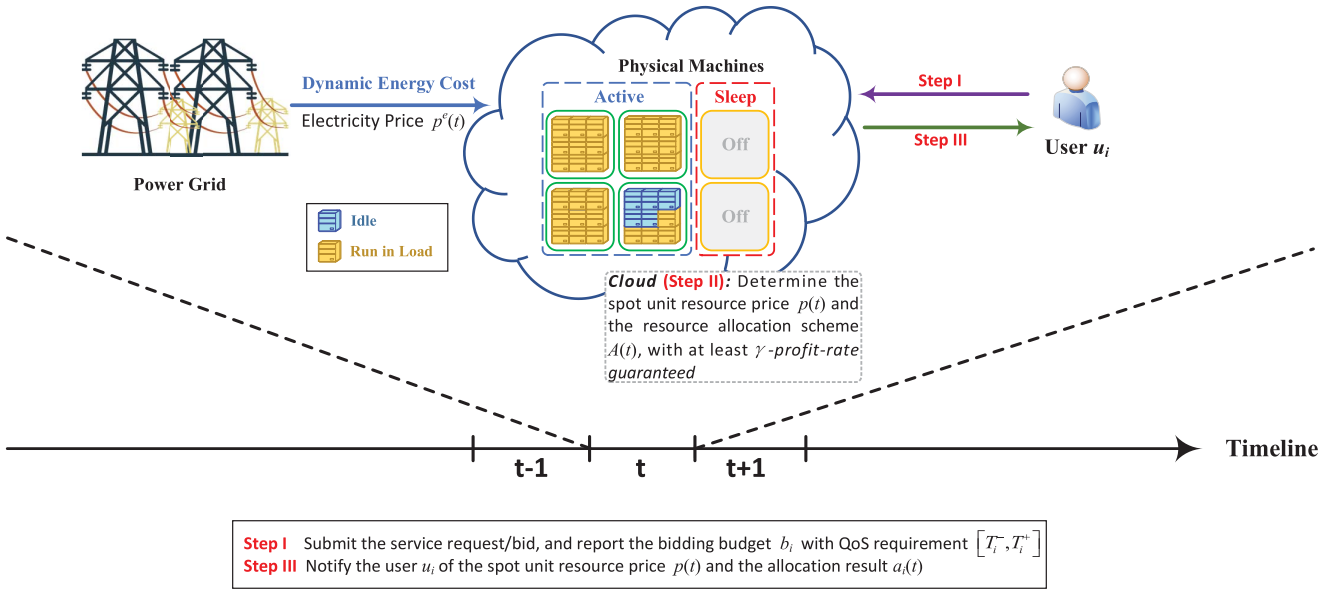


Fig. 1. Overview of Cloud Ecosystem and Resource Auction Framework.

the social welfare over various users requesting for cloud resources (i.e., an envy-free status), through granting varied priorities to distinct users during cloud resource auction. Yang *et al.* [39] deployed a consensus revenue estimate strategy in cloud resource auction which enabled the envy-freeness on heterogeneous VM allocation.

In this article, we intend to consolidate the research of market-based cloud pricing strategy from the perspective of resource auction. Different from [27], [29], [31], [33] involving competitive cloud service providers in the auction, we focus on maximizing a cloud service provider's utility, such that the maximum users are incentivized to be served by the cloud service provider while a minimum profit rate is guaranteed. Besides, compared with [28], [29], [31]–[33] using discriminatory price auction, we adopt a single-price auction where cloud resources are valued in the same unit price. The single-price auction can be more widely accepted, because of the common sense that identical goods should be identically valued. Moreover, our price-incentive resource auction mechanism also satisfies the essential properties of budget balance, truthfulness and envy-freeness. It is useful to enhance the robustness of resource auction mechanism.

III. System Design and Formulation

A. System Model

High-level Overview: We consider our cloud ecosystem which works at a time-slotted manner, and the time horizon is discretized into time slots at the duration of τ, indexed by t. Resource auction is conducted over time slots. At the beginning of each time slot, each user arbitrarily determines whether to make the cloud service request. If the service request is proposed, the user would place the service bid with her QoS requirement to the cloud service provider. At each time slot, the cloud service provider aggregates the service bids, and then decides the spot unit resource price and the resource allocation scheme across multiple service bids. In response to the dynamic service bids and energy costs, results of resource pricing and allocation are stipulated to be reevaluated at each time slot. The cloud service provider would update its decision on resource pricing and allocation, based on the latest service bids and energy costs.

Fig. 1 outlines our cloud ecosystem, where the interaction between the cloud service provider and users is presented. We identify the resource auction process at each time slot t into three steps. In Step I, the users who place service bids at the time slot t inform the cloud service provider of their QoS requirements and bidding budgets. In Step II, the cloud service provider receives the service bids, and determines the spot unit resource price $p(t)$ and resource allocation scheme during the time slot t. With the spot unit resource price of $p(t)$, the cloud service provider decides for each user u_i the $a_i(t)$ units of resources to be allocated and bundled as a customized VM where the user u_i is served during the time slot t. The decision on resource pricing and allocation is made based on a collective consideration of energy costs and user bids. To reduce nonessential energy costs and increase the profit earning, idle physical machines would be switched off into the sleep mode. In Step III, the cloud service provider notifies the resource price and allocation result of each user who places the service bid, and collects the corresponding service payment.

Cloud Users: There are totally N cloud users who propose the cloud service request, denoted by $\mathcal{U} = \{u_1, \ldots, u_N\}$. We assume that, each user u_i proposes a single service request, and the user who proposes multiple service requests can be regarded as a group of users. The service request of user u_i is specified as a tuple $(s_i, t_i^-, t_i^+, T_i^-, T_i^+, b_i)$. The service type s_i labels the type of application requested for execution at cloud. As in [34], [40], we mainly study on the computation-intensive applications served at the cloud. The temporal interval $[t_i^-, t_i^+]$ is subscribed by the user u_i to perform the application with the type of s_i, where $t_i^- \in \mathbb{N}$ and $t_i^+ \in \mathbb{N}$ respectively indicate the corresponding starting time

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Definition
τ	Duration of a time slot.
t	Index of a time slot.
$\mathcal{U}$	The set of users $\langle u_i \rangle_{i=1}^{N}$ making service requests over time slots.
$\mathcal{U}(t)$	The subset of users making service requests during the time slot t.
N	Number of users making cloud service requests.
$N(t)$	Number of users placing service bids during the time slot t.
r	Resource capacity of each physical machine at the cloud.
$\widetilde{c}$	Average energy cost to run a physical machine per time slot.
$p(t)$	Spot price per unit of cloud resource at the time slot (i.e., billing cycle) t.
$p^e(t)$	Electricity price during the time slot t.
γ	Minimum profit rate required by the cloud service provider, i.e., ratio of profit over cost.
s_i	Service type of user u_i, i.e., the type of application requested by user u_i for execution at cloud.
$[t_i^-, t_i^+]$	Temporal interval subscribed by user u_i to perform the application at cloud.
T_i^-	Optimal job runtime of user u_i.
T_i^+	Slowest acceptable job runtime of user u_i.
b_i	Bidding budget of user u_i per time slot (billing cycle).
a_i^+	Maximum resource demand of user u_i enabling the job runtime T_i^-.
a_i^-	Minimum resource demand of user u_i enabling the job runtime T_i^+.
$a_i(t)$	Number of cloud resource units purchased by user u_i at the time slot t.
$A(t)$	Resource allocation scheme at the time slot t, i.e., $A(t) \leftarrow \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$.
$T_i(a_i(t))$	Job Runtime gained with $a_i(t)$ units of cloud resources assisted.
$u_i(a_i(t))$	Utility of user u_i as a function of $a_i(t)$.
$v_i(a_i(t), p(t))$	Net utility of user u_i as a binary function of $a_i(t)$ and $p(t)$.

slot and the ending time slot. Since the cloud service provider conducts resource pricing and auction by time slots, thus the user u_i would automatically place her service bid at the beginning of each time slot between $[t_i^-, t_i^+]$ to compete for cloud resources, where b_i indicates the bidding budget per time slot. At each time slot *t*, a subset of users $\mathcal{U}(t) \subset \mathcal{U}$ place the service bid to the cloud service provider. Let *N*(*t*) denote the number of users placing the bid at the time slot *t*.

During the temporal interval $[t_i^-, t_i^+]$, the user u_i would execute a sequence of jobs whose service type is s_i. The runtime to complete a single job is differentiated with the service type s_i and the number of allocated cloud resources. According to the regression analysis results in [12] and the experimental evidence in [41], the runtime to complete a single job for user u_i can be estimated by (1).

$$T_i(a_i(t)) = \theta_{s_i} + \phi_{s_i} \cdot a_i(t) + \frac{\eta_{s_i}}{a_i(t)} + \omega_{s_i} \cdot \log(a_i(t)) \quad (1)$$

where θ_{s_i}, ϕ_{s_i}, η_{s_i} and ω_{s_i} are regression coefficients specific to the service type s_i, and $a_i(t)$ represents the number of cloud resources allocated to user u_i during the time slot *t*.

Each user u_i can be served in her own customized VM [5] assembled by the user-specified amount (i.e., $a_i(t)$) of cloud resources, as in some recently burgeoning cloud platforms [42], [43]. Such a customized VM service is beneficial to attract more cloud users. Meanwhile, the cloud resource can be in various forms, including CPU units, memory size, disk storage, network bandwidth, etc. Enlightened by the *degree of parallelism* specified by Spark/Hadoop [44], the amount of cloud resources $a_i(t)$ for allocation can represent the number of allocated compute slots, each of which is assembled by a fixed number of varied resources.

Within the service request, the user u_i also reports her QoS requirement as $[T_i^-, T_i^+]$, where T_i^-, $T_i^+ \in \mathbb{R}^+$. It states the job runtime that the cloud service provider commits to, if the user u_i's service bid is accepted. Specifically, it should take the time no more than T_i^+ to complete a user u_i's job. Besides, the job runtime which is less than T_i^- is needless for the user u_i. Let a_i^+ and a_i^- represent the amount of cloud resources which just respectively enables a job accomplished within the time of T_i^- and T_i^+, i.e.,

$$a_i^+ = T_i^{-1}(T_i^-) \quad \text{and} \quad a_i^- = T_i^{-1}(T_i^+).$$

Based on the above, the user u_i has an utility $u_i(a_i(t))$ which indicates her "happiness" towards various resource allocation results $a_i(t)$, formulated by the piecewise function (2) below.

$$u_i(a_i(t)) = \begin{cases} \rho_i(a_i^+) & \text{if } a_i(t) \in (a_i^+, +\infty) \\ \rho_i(a_i(t)) & \text{if } a_i(t) \in [a_i^-, a_i^+] \\ 0 & \text{if } a_i(t) \in [0, a_i^-) \end{cases} \quad (2)$$

When the allocated cloud resources $a_i(t) \in [a_i^-, a_i^+]$, the user utility $u_i(a_i(t))$ is evaluated by the QoS progress rate $\rho_i(a_i(t))$ relative to T_i^+, as formulated in (3). It is worth stressing that the QoS progress rate is by no means proportional to the allocated resources, but usually with concavity [41].

$$\rho_i(a_i(t)) = \frac{T_i^+}{T_i(a_i(t))}. \quad (3)$$

Cloud Service Provider: We restrict our focus on a homogeneous set of physical machines managed by the cloud service provider, as in [26], [45]. Meanwhile, the cloud service provider generally promises to supply sufficient computational resources, and the recent studies [46], [47] also indicate that the resource utilization in large-scale data centers is lower than 50% most of the time. Thus, we assume that the cloud service provider has infinite resource capacity, where each physical machine of cloud is equipped with *r* units of computational resources. The cloud service provider controls the sleep/active state of each physical machine over time slots, where the active

physical machines are supplied to users who make cloud service requests. At each time slot (i.e., billing cycle) *t*, the cloud resources are priced by units as the spot unit resource price *p*(*t*), and $a_i(t)$ units of cloud resources are allocated for each user $u_i \in \mathcal{U}(t)$ during the time slot *t*. Therefore, as in [20], the number of active physical machines needed during the time slot *t* is estimated by (4).

$$m(t) = \left\lceil \frac{\sum_{u_i \in \mathcal{U}(t)} a_i(t)}{r} \right\rceil \tag{4}$$

Let $\widetilde{c}$ indicate the average energy cost to operate a physical machine per time slot, and thus the overall energy cost to perform *m*(*t*) physical machines during the time slot *t* is $\widetilde{c} \cdot m(t)$. The electricity price is dynamically fluctuated over different time slots [21], [22], denoted by $p^e(t)$ which is reasonably well informed at the beginning of each time slot *t*. Hence, the energy bill of the cloud service provider during the time slot *t* is expressed as $\widetilde{c} \cdot p^e(t) \cdot m(t)$. In addition, at the time slot *t*, the cloud service provider collects the service payment from users, which is $p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t)$ in total.

A minimum profit rate $\gamma > 0$ is arbitrarily pre-defined by the cloud service provider according to its profitability objective. The cloud service provider makes the decision on *p*(*t*), $a_i(t)$ and *m*(*t*) with the minimum profit rate of γ guaranteed, which implies the constraint (5) below.

$$p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) \geq (1+\gamma) \cdot \widetilde{c} \cdot p^e(t) \cdot m(t). \tag{5}$$

Charge: The user u_i has a bidding budget $b_i \in \mathbb{R}^+$, representing the maximum willing-to-pay monetary expense at each time slot (i.e., billing cycle). The service payment $(p(t) \cdot a_i(t))$ of user u_i at each time slot should not exceed the bidding budget b_i. Then, the net-utility $v_i(a_i(t), p(t))$ of user u_i is evaluated by the difference between her utility $u_i(a_i(t))$ and her service payment over her allocated cloud resources $a_i(t)$, such that

$$v_i(a_i(t), p(t)) = u_i(a_i(t)) - \frac{p(t) \cdot a_i(t)}{b_i} \tag{6}$$

Note that the service payment of user u_i is normalized by her bidding budget b_i, which implies that the service payment $(p(t) \cdot a_i(t))$ should not exceed the bidding budget b_i. By integrating (2) and (6) into account, the net-utility $v_i(a_i(t), p(t))$ can be also expressed as (7).

$$v_i(a_i(t), p(t)) = \begin{cases} \rho_i\left(a_i^+\right) - \frac{p(t) \cdot a_i(t)}{b_i} & \text{if } a_i(t) \in \left(a_i^+, +\infty\right) \\ \rho_i(a_i(t)) - \frac{p(t) \cdot a_i(t)}{b_i} & \text{if } a_i(t) \in \left[a_i^-, a_i^+\right] \\ -\frac{p(t) \cdot a_i(t)}{b_i} & \text{if } a_i(t) \in \left[0, a_i^-\right). \end{cases} \tag{7}$$

Remark: During the subscription temporal interval $[t_i^-, t_i^+]$, the user u_i can arbitrarily adjust her bidding budget based on the dynamic status of resource competition, to maintain its resource competitive edge. By a slight abuse of notations, we use b_i to represent the bidding budget rather than $b_i(t)$.

B. Optimization Problem

We formulate the resource pricing and auction problem from the standpoint of the cloud service provider which looks after the interests of both cloud service provider and users. It targets at stimulating maximum users outsourcing their application to cloud while the minimum profit rate of γ is ensured as well.

Before formulating our optimization problem, we firstly investigate the resource demand of user which reacts to the cloud pricing setting. In order to incentivize the user u_i to conduct cloud migration, the user u_i is supposed to gain the maximum net-utility over her cloud resource allocation and payment. Let $d_i(p(t))$ represent the user u_i's resource demand (in terms of the amount of resources) under the spot unit resource price *p*(*t*), where the user u_i's net-utility is maximized. We can formulate the user u_i's resource demand under the spot unit resource price *p*(*t*) as:

$$d_i(p(t)) \triangleq \arg\max_{a_i(t)} \; v_i(a_i(t), p(t)) \tag{8}$$

With the resource demand $d_i(p(t))$ of each user defined, our resource pricing and auction problem can be formulated as Problem P1. The resource price *p*(*t*) and the resource allocation scheme $A(t) = \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$ are a couple of decision variables correlatively solved at each time slot *t*, where the resource allocation scheme *A*(*t*) is resolved based on user demands under a determined resource price *p*(*t*).

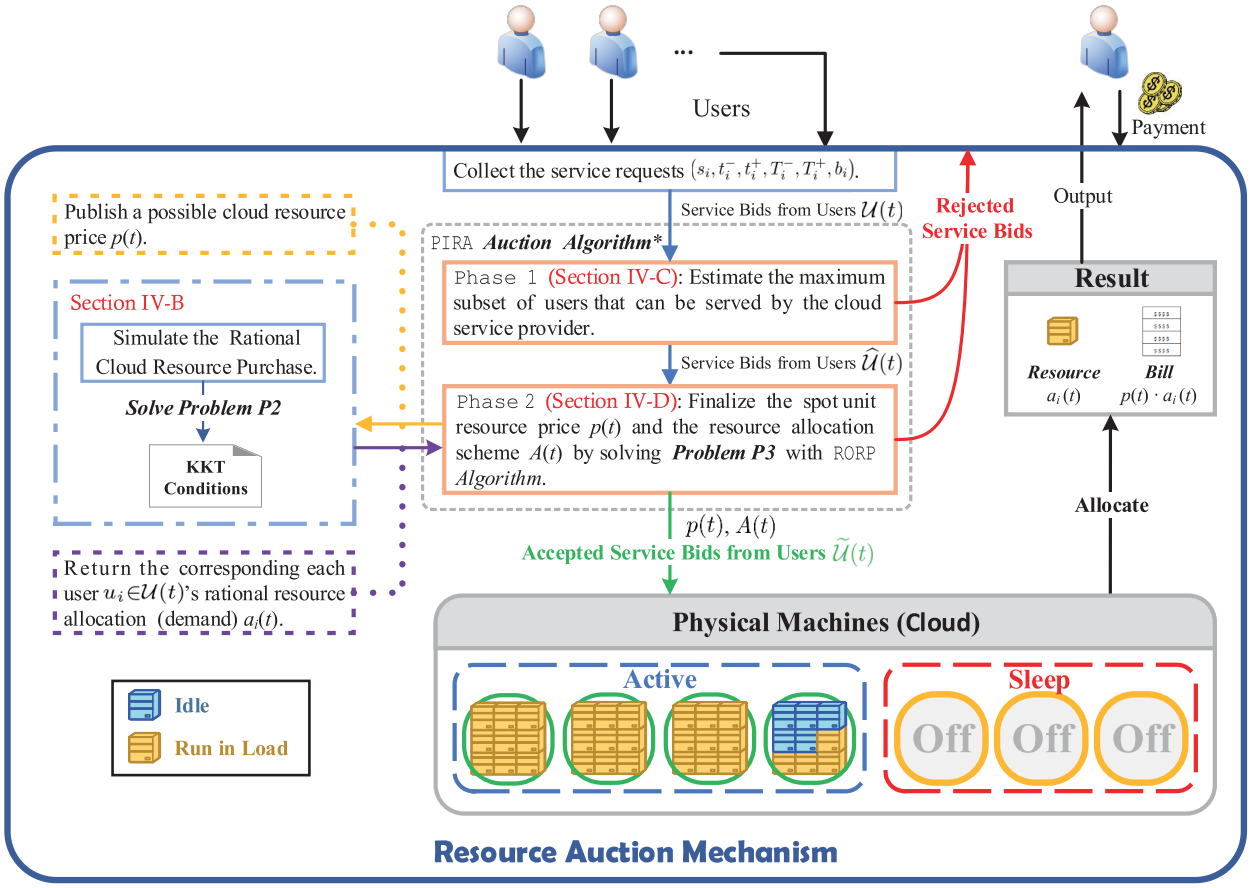
$$\max_{a_i(t), p(t)} \sum_{u_i \in \mathcal{U}(t)} I_{\{a_i(t) > 0\}} \tag{P1}$$

$$\text{s.t.} \quad p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) \geq (1+\gamma) \cdot \widetilde{c} \cdot p^e(t) \cdot m(t) \tag{C1.1}$$

$$a_i(t) = d_i(p(t)) \quad \forall u_i \in \mathcal{U}(t) \tag{C1.2}$$

The objective of Problem P1 maximizes the number of users served at cloud, where $I_{\{a_i > 0\}}$ is the indicator function returning 1 when $a_i > 0$, otherwise 0. Constraint C1.1, which is derived from (5), claims the minimum profitability objective of the cloud service provider. Constraint C1.2 indicates each user purchases cloud resources on a voluntary basis, where the amount of purchased cloud resources equals to her demand.

Solving the above resource pricing and auction problem is non-trivial due to the following three aspects. Firstly, Problem P1 maps to the bin packing problem, which is NP-hard. The physical machines at cloud are conceived as bins, and each user's resource demand regarded as the object requires to be packed into the bins. Resulting from the NP-hardness of bin packing problem, Problem P1 is NP-hard as well. The second challenge results from the concavity and segmentation of the net-utility function $v_i(\cdot)$. In Problem P1, the amount of purchased cloud resources $a_i(t)$ is determined for each user u_i based on her own net-utility maximization, which complicates the problem solving. The last challenge is embodied in the complex correlation between the two families of decision variables *p*(*t*) and $a_i(t)$. The decision on *p*(*t*) determines the resource allocation (demand) $a_i(t)$. As a return, the resource allocation scheme $A(t) = \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$ decides the



* PIRA auction algorithm solves ***Problem P1*** with a decomposition method respectively in Phase 1 and Phase 2.

Fig. 2. Overview of Price-Incentive Resource Auction Mechanism.

optimization result $\sum_{u_i \in \mathcal{U}(t)} I_{\{a_i > 0\}}$, and is closely related to whether the cloud service provider's minimum profit rate γ is satisfied under the price setting of $p(t)$. Given the above, we need to design a resource auction mechanism to solve this problem in an efficient and effective manner.

Remark: Although the objective of our optimization problem is not to maximum the cloud service provider's revenue, the formulation of Problem P1 has practical relevance from the perspective of the cloud service provider. *First*, from the viewpoint of cloud ecosystem, our Problem P1 is aimed to motivate the maximum users served at cloud, which is contributive for the cloud service provider to gain its market share in the cloud marketplace. *Second*, we consider the scenario where the user volunteers to purchase a certain amount of cloud resources maximizing her net-utility. Compared with a pure pursuit of revenue maximization regardless of the user interests, our scenario is much closer to the realistic cloud marketplace. *Third*, a minimum profit rate γ is defined to ensure the interest of cloud service provider. The minimum profit rate γ can be treated as the parameter taking the trade-off between the interests of users and the cloud service provider.

IV. Mechanism Design

A. Mechanism Overview

Our resource auction mechanism is aimed at incentivizing the maximum cloud users while a minimum profit rate γ should be guaranteed, as formulated in Problem P1. Problem P1 belongs to the bin packing problem, which has non-polynomial computation complexity. Given this, we shift our focus on designing an efficient resource auction mechanism for Problem P1.

Fig. 2 outlines our price-incentive resource auction mechanism. We first need to clarify the rational strategy of cloud resource purchase, exploring how many cloud resources each user demands for under a certain determined resource price. According to our optimization problem P1, each user purchases the amount of cloud resources according to her demand with her own net-utility maximized. The amount of required cloud resources should be efficiently solved out, but the segmentation and concavity of the net-utility function $v_i(\cdot)$ bring computational challenges. In view of this, we extensively study the rational strategy of cloud resource purchase in Section IV-B.

While the rational cloud resource purchase strategy concerns about how each user purchases resources achieving her own net-utility maximization, our cloud resource auction algorithm is price-incentive and focuses on pricing the cloud resource from the high level. Each user's resource demand/allocation can be effectively regulated. The detailed design for our resource auction algorithm is introduced in Sections IV-C and IV-D. In brief, our price-incentive resource auction algorithm is identified by two main phases to make

the decision on resource pricing and allocation, summarized as follows.

- *In Phase* 1, *the maximum subset of users who can be served at cloud is estimated in a crude but efficient manner.*
  - The cloud service provider should satisfy the QoS requirement for the maximum users, without breaking the *price floor* $\lfloor p \rfloor$ (detailed later). Based on this criteria, the estimated maximum cloud user subset $\widehat{\mathcal{U}}(t)$ is obtained.
- *In Phase* 2, *the spot unit resource price p(t) and resource allocation scheme A(t) are further finalized.*
  - Given the estimation result $\widehat{\mathcal{U}}(t)$ in Phase 1, the cloud service provider further finalizes the maximum cloud user subset $\widetilde{\mathcal{U}}(t)$, where the minimum profit rate γ is assuredly guaranteed.
  - The spot unit resource price *p*(*t*) is also optimized which helps to earn as much revenue as possible. The corresponding resource allocation scheme $A(t) = \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$ is obtained via simulating each user's rational cloud resource purchase.

B. Rational Cloud Resource Purchase Strategy

We intend to clarify how many cloud resources are demanded by each user under a certain determined resource price. It is assumed that the spot unit resource price has been determined as *p*(*t*). As expressed in (8), each user u_i determines her resource demand $d_i(p(t))$ by solving a net-utility maximization problem. Indicated by (6) and (7), the user u_i's net-utility $v_i(a_i(t), p(t))$ is jointly associated with the QoS gain and the service payment, and varies with the amount of allocated resources $a_i(t)$. To obtain the optimal $a_i(t)$ maximizing the net-utility, we divide the following two cases into discussion.

- *Case 1:* The user u_i stretches her budget b_i to purchase the maximum amount of cloud resources which is less than a_i^-.
  - The user u_i's QoS requirement $[T_i^-, T_i^+]$ cannot be satisfied, where the net-utility $v_i(a_i(t), p(t))$ decreases with $a_i(t)$. The user u_i would gain a negative net-utility if the user u_i purchases cloud resources. Thus, the user u_i tends to be rejected at the time slot *t*, and abandons cloud migration with $a_i(t) = 0$.
- *Case 2:* The user u_i has a budget to purchase $a_i(t) > a_i^-$ units of cloud resources with her QoS requirement satisfied.
  - We first demonstrate that the user u_i could not purchase the amount of cloud resources greater than a_i^+. Although the user u_i may afford to purchase $a_i(t) > a_i^+$ units of cloud resources, the amount of purchased cloud resources over a_i^+ is needless and merely results in disutility instead, because the user u_i's QoS gain has been overfulfilled beyond her requirement T_i^-. Hence, purchasing $a_i(t) > a_i^+$ units of cloud resources contradicts with the objective of net-utility maximization for the user u_i.
  - Thus, the optimal $a_i(t)$ maximizing the user u_i's net-utility should be figured out within $[a_i^-, a_i^+]$. According to (7), finding out the optimal $a_i(t)$ is to balance the trade-off between the QoS gain and the service payment. However, since the QoS progress function $\rho_i(a_i(t))$ (see Eq. (3)) representing the QoS gain is concave in general, it is intractable to solve out the optimal $a_i(t)$ which maximizes the user u_i's net-utility.

From now on, we get started on attacking the intractability to solve the optimal $a_i(t)$ for the *Case* 2. Within $a_i(t) \in [a_i^-, a_i^+]$, the user u_i would gain much more QoS progress $\rho_i(a_i(t))$ with the increase of $a_i(t)$ while a higher service payment is charged in the meantime. Since the QoS progress function $\rho_i(a_i(t))$ is a concave function, thus the optimal $a_i(t)$ can be a *breakeven resource allocation point* where the growth rate of QoS progress $\rho_i'(a_i(t))$ exactly counteracts the increase rate of normalized service payment $p(t)/b_i$. If such a *breakeven point* doesn't exist, that is to suggest $\rho_i'(a_i^-) \leq p(t)/b_i$. Here, the optimal $a_i(t)$ should be a_i^-. To summarize, the unique optimal solution for $a_i(t)$ is admitted as:

$$\begin{cases} a_i(t) = a_i^- & \text{if } \rho_i'(a_i^-) \leq p(t)/b_i \\ \rho_i'(a_i(t)) = p(t)/b_i & \text{if } \rho_i'(a_i^-) > p(t)/b_i \end{cases} \tag{9}$$

After analyzing the unique existence of the optimal $a_i(t)$, we expect to propose a computational-efficient approach to solve out the optimal $a_i(t)$ for the user u_i. Here, we formulate the net-utility maximization problem P2 which can figure out the optimal $a_i(t)$. Constraint 2.1 ensures the budget balance, meaning that the service payment should not exceed the bidding budget. Constraint C2.2 specifies the feasible range of $a_i(t) \in [a_i^-, a_i^+]$.

$$\max_{a_i(t)} \ \rho_i(a_i(t)) - \frac{p(t) \cdot a_i(t)}{b_i} \tag{P2}$$

$$\text{s.t.} \ \ p(t) \cdot a_i(t) \leq b_i \tag{C2.1}$$

$$a_i(t) \in \left[a_i^-, a_i^+\right] \tag{C2.2}$$

It can be easily identified that Problem P2 is a convex optimization problem, in accordance with the standard form of convex optimization problem [48] (The proof is omitted). Then, we can solve the convex optimization problem P2 through searching for the solution satisfying the Karush-Kuhn-Tucker (KKT) conditions [48]. Henceforth, we can acquire the optimal $a_i(t)$ maximizing the user u_i's net-utility for the *Case* 2, by searching for the solution which satisfies the KKT conditions of Problem P2.

Briefly speaking, we epitomize the rational cloud resource purchase strategy of each user when a certain resource price *p*(*t*) has been determined. In the *Case* 1, the user u_i abandons cloud migration with $a_i(t) = 0$. In the *Case* 2, the user u_i determines the amount of purchased cloud resources $a_i(t)$ by solving the KKT conditions of Problem P2.

Computation Complexity Analysis: In the rational cloud resource purchase strategy, we mainly focus on the computation complexity in the *Case* 2 including the KKT-condition

solving. Generally, the computation complexity of solving the KKT conditions exponentially increases with the number of inequation constraints in the convex optimization problem [49]. Since there are $K = 3$ inequation constraints (i.e., the upper bound of Constraint C2.1, and the upper and lower bound of Constraint C2.2) in Problem P2, then we can solve Problem P2 for the user u_i by searching for the solution which satisfies the KKT conditions at the computation complexity of $O(2^K) = O(1)$. Given this, with a spot unit resource price $p(t)$ determined, we sequentially calculate the resource demand $a_i(t) = d_i(p(t))$ of each user $u_i \in \mathcal{U}(t)$ by solving the corresponding Problem P2, requiring the overall computation complexity of $O(N(t))$. Further, Problem P2 of each user u_i is independent with no correlation, therefore the resource demand of each user $u_i \in \mathcal{U}(t)$ can be also solved in parallel, in the case of which the computation complexity is reduced to $O(1)$.

Remark: Because of the space limitation, the more detailed formulation and analysis of KKT conditions for Problem P2 is presented in Appendix A of the supplementary material.

C. Phase 1 - Estimate the Maximum Subset of Cloud Users

At the beginning of each time slot t, the cloud service provider collects the service bids proposed by the users $u_i \in \mathcal{U}(t)$. Each user $u_i \in \mathcal{U}(t)$ has the heterogeneous bidding budget b_i and various resource requirement $[a_i^-, a_i^+]$. We can obtain each user u_i's maximum acceptable price p_i^+ for unit cloud resource, formulated by (10). If the spot unit resource price $p(t)$ is set higher than p_i^+, then the user u_i cannot afford to purchase at least a_i^- units of cloud resources to meet her QoS requirement, and hence abandons cloud migration with $a_i(t) = 0$.

$$p_i^+ = \frac{b_i}{a_i^-} \quad \text{for the user } u_i \in \mathcal{U}(t) \tag{10}$$

We order the users $u_i \in \mathcal{U}(t)$ based on p_i^+ in an ascending rank $\Theta = < u_{[1]}, u_{[2]}, \ldots, u_{[N(t)]} >$, where $u_{[i]}$ indicates the user ranked at the i-th place in Θ. If $p(t) \leq p_{[i]}^+$, then the users $u_{[j]}$ ($j \geq i$) could purchase cloud resources with their QoS requirements fulfilled. But if $p(t) > p_{[i]}^+$, then the users $u_{[j]}$ ($j \leq i$) are unaffordable to purchase enough cloud resources meeting their QoS requirements, hence having to abandon cloud migration.

In order to estimate the maximum subset of users served at cloud, we consider the minimum profit rate γ required by the cloud service provider. Here, we put forward a *price floor* as $\lfloor p \rfloor = \widetilde{c} \cdot p^e(t) \cdot (1+\gamma)/r$, which is used to compare with each user $u_{[i]}$'s maximum acceptable price $p_{[i]}^+$. The *price floor* indicates the unit resource price which equally splits a physical machine's bottom price $\widetilde{c} \cdot p^e(t) \cdot (1+\gamma)$. If $p_{[i]}^+ < \lfloor p \rfloor$ for the user $u_{[i]}$, then the service bid of user $u_{[i]}$ cannot be accepted by the cloud service provider at the time slot t. In this phase, we intend to filter out all the users $u_{[i]}$ whose $p_{[i]}^+ < \lfloor p \rfloor$ in the order Θ. Specifically, the critical user $u_{[i'']}$ amongst users should be figured out, satisfying that

$$p_{[i'']}^+ \geq \lfloor p \rfloor \quad \text{and} \quad p_{[j]}^+ < \lfloor p \rfloor \ (j < i'') \tag{11}$$

In other words, the service bids of users $u_{[j]}$ ($j < i''$) are rejected. In the end, an estimation of the maximum cloud user subset is presented as

$$\widehat{\mathcal{U}}(t) = \langle u_{[i'']}, u_{[i''+1]}, \ldots, u_{[N(t)]} \rangle. \tag{12}$$

D. Phase 2 - Finalize the Spot Unit Resource Price and Resource Allocation Scheme

Based on the estimation result in Phase 1, we further finalize the maximum cloud user subset, together with the spot unit resource price and resource allocation scheme determined as well. Before introducing the details, we preliminarily present the resource price spectrum which enables various cloud user subsets. Let $[p(t)^-, p(t)^+]$ represent the resource price spectrum of $p(t)$ which could exactly empower the users $\{u_{[i]}, u_{[i+1]}, \ldots, u_{[N(t)]}\}$ willing to be served at cloud. Recall that, the maximum acceptable price for the user $u_{[i]}$ is $p_{[i]}^+$, as formulated in (10). If the spot unit resource price $p(t) \leq p_{[i]}^+$, then the users including $u_{[j]}$ ($j = i, i+1, \ldots, N(t)$) would like to purchase cloud resources with their QoS requirements satisfied. Therefore, the resource price spectrum $[p(t)^-, p(t)^+]$ is defined as follows.

$$p(t)^+ = p_{[i]}^+$$
$$p(t)^- = \begin{cases} \frac{\widetilde{c} \cdot p^e(t) \cdot (1+\gamma)}{r} & \text{if } i = 1 \\ p_{[i-1]}^+ + \sigma & \text{otherwise} \end{cases}$$

where $\sigma > 0$ can be a minuscule positive number just making $p_{[i-1]}^+ + \sigma > p_{[i-1]}^+$. Meanwhile, $p(t)^- = \lfloor p \rfloor$ when $i = 1$.

It is worth noting that, the *price floor* $\lfloor p \rfloor$ (defined in Phase 1) is no more than a necessary condition ensuring the minimum profit rate γ. Let us consider the following scenario, where $\sum_{u_i \in \mathcal{U}(t)} a_i(t)/r = 50.3$ indicates that $m(t) = 51$. Here, the cost plus the required minimum profit over the $0.7r$ units of idle cloud resources should be additionally shared by the users who purchase cloud resources. It follows that, the *price floor* $\lfloor p \rfloor$ is insufficient to provide the γ-profit-rate guarantee.

Therefore, we need to further ascertain the maximum cloud user subset based on the rough estimation in Phase 1. The basic idea of finalizing the maximum cloud user subset is to gradually downscale the subset of cloud users until the minimum profit rate γ is assuredly achieved. In specific, we initially determine the cloud user subset as our estimated maximum cloud user subset $\widehat{\mathcal{U}}(t)$ (i.e., $\{u_{[i'']}, ..., u_{[N(t)]}\}$). Here, if the minimum profit rate γ can be achieved at a resource price between the corresponding resource price spectrum $[p(t)^-, p(t)^+]$, then the estimated maximum cloud user subset in Phase 1 is literally the maximum cloud user subset, i.e., $\widetilde{\mathcal{U}}(t) \leftarrow \widehat{\mathcal{U}}(t)$. Otherwise, we downsize the cloud user subset by rejecting the service request of user $u_{[i'']}$, and then similarly testify whether it is able to guarantee the minimum profit rate γ in this time. If not, the cloud user subset would be further downsized by removing the user $u_{[i''+1]}$. The above

process would not quit the loop until the minimum profit rate γ is assured satisfied, or rejecting all proposed service bids. The final maximum subset of cloud users is determined as $\widetilde{\mathcal{U}}(t)$.

When finding out the unit resource price $p(t)$ between $[p(t)^-, p(t)^+]$ which makes the minimum profit rate γ satisfied, it is preferable to figure out the optimal unit resource price which achieves the higher revenue for the cloud service provider. Therefore, the revenue maximization problem is formulated by Problem P3, where the cloud service provider's revenue π is defined in C3.1. Problem P3 not only specifies the resource price spectrum $p(t) \in [p(t)^-, p(t)^+]$ as shown in C3.2, but also has the same constraints as Problem P1.

$$\max_{a_i(t), p(t)} \quad \pi \tag{P3}$$

$$\text{s.t.} \quad \pi = p(t) \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) \tag{C3.1}$$

$$p(t)^- \le p(t) \le p(t)^+ \tag{C3.2}$$

$$(\text{C1.1}) - (\text{C1.2})$$

Note that, in Constraint C1.2, the segmentation and concavity of the net-utility function $v_i(\cdot)$ makes our revenue maximization problem intractable. Enlightened by the unique structure of this optimization problem, however, we put forward a simple but effective Revenue-Optimal Resource Pricing algorithm namely RORP which reaches a near-optimal revenue. The pseudocode of the RORP algorithm is shown in Algorithm 1. It mainly consists of two steps as follows.

- *Step 1*: Discretize the continuous resource price spectrum $[p(t)^-, p(t)^+]$ into X candidate prices $\hat{p}_x$, $x = 1, 2, \ldots, X$ (Line 1-3).
- *Step 2*: Find out the candidate price $\hat{p}(t) = \hat{p}_x$ amongst X candidate prices, under which the maximum revenue is gained (Line 4-14).

Technically, the resource price spectrum $[p(t)^-, p(t)^+]$ is discretized into X discrete candidate prices, where X is calculated based on (13).

$$X = \left\lfloor \frac{\log\left(p(t)^+ / p(t)^-\right)}{\log(1+\epsilon)} \right\rfloor + 1 \tag{13}$$

Let $\hat{p}_x$ denote the x-th candidate price, which is expressed in (14). Note that, $\epsilon > 0$ is a constant parameter which is configured according to the required trade-off between computation efficiency and approximation ratio. A smaller ϵ implies a better approximation ratio obtained by RORP algorithm, which also requires a higher price sampling intensity over $[p(t)^-, p(t)^+]$ (i.e., a greater X) incurring a greater computation complexity (detailed later in Theorem 1).

$$\hat{p}_x = p(t)^+ \cdot (1+\epsilon)^{1-x} \tag{14}$$

The RORP algorithm respectively calculates the revenue π associated with each candidate price $\hat{p}_x$. After that, the candidate price $\hat{p}(t) = \hat{p}_x$ gaining the maximum revenue would be selected as the final spot unit resource price $p(t)$. The corresponding resource allocation scheme $A(t) = \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$ is acquired through simulating each user's rational cloud resource purchase under $p(t) = \hat{p}(t)$.

Algorithm 1: Revenue-Optimal Resource Pricing Algorithm (RORP)

Input: Resource price spectrum $\left[p(t)^-, p(t)^+\right]$.
Output: Resource price $p(t)$;
Resource allocation scheme $A(t)$.

1 Initialize $\pi \leftarrow 0$, $p(t) \leftarrow$ null, $A(t) \leftarrow$ null;
2 Calculate the number of discrete prices X based on (13);
3 Discretize the continuous price spectrum $\left[p(t)^-, p(t)^+\right]$ into X prices as $\hat{p}_x \leftarrow p(t)^+ \cdot (1+\epsilon)^{1-x}, x = 1, 2, \ldots, X$;
4 **for** *each* $x = \{1, 2, \ldots, X\}$ **do**
5 **for** *each* $u_i \in \mathcal{U}(t)$ **do**
6 **if** $p_i^+ \le \hat{p}_x$ **then**
7 Acquire the user u_i's rational resource allocation (demand) $a_i(t) = d_i(\hat{p}_x)$ solving the KKT conditions of Problem P2;
8 **else**
9 $a_i(t) \leftarrow 0$;
10 **if** $\frac{\hat{p}_x \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t)}{(1+\gamma)} \ge \widetilde{c} \cdot p^e(t) \cdot \left\lceil \frac{\sum_{u_i \in \mathcal{U}(t)} a_i(t)}{r} \right\rceil$ **and** $\hat{p}_x \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t) > \pi$ **then**
11 $\pi \leftarrow \hat{p}_x \cdot \sum_{u_i \in \mathcal{U}(t)} a_i(t)$;
12 $p(t) \leftarrow \hat{p}_x$;
13 $A(t) \leftarrow \langle a_i(t) \rangle_{u_i \in \mathcal{U}(t)}$;
14 **return** $p(t)$ and $A(t)$;

TABLE II
APPROXIMATION RATIO VS. COMPUTATIONAL EFFICIENCY

Approximation Ratio $1+\epsilon$	1.02	1.03	1.04	1.05	1.10	1.20
Number of Discrete Prices X	117	48	59	48	25	13

* Exemplified by $p(t)^+ / p(t)^- = 10$.

Theorem 1: The RORP algorithm can achieve an approximation ratio of $(1+\epsilon)$ on cloud service provider's revenue maximization, together with the computation complexity of $O(X)$ supposing that $N(t)$ users conduct the rational cloud resource purchase in parallel.

Proof: Suppose the revenue-optimal spot unit resource price is $p^*(t)$, under which the cloud service provider gains the optimal revenue as π^*. There must exist an integer $y \in [1, X]$ such that $p(t)^+ \cdot (1+\epsilon)^{-y} \le p^*(t) \le p(t)^+ \cdot (1+\epsilon)^{1-y}$. In our RORP algorithm, $\hat{p}(t)$ is the approximate solution selected from the X candidate discrete prices, under which the corresponding revenue is $\hat{\pi}$. Here, we can obtain that

$$\begin{aligned}
\pi^* &= p^*(t) \times \sum_{u_i \in \mathcal{U}(t)} d_i(p^*(t)) \\
&\le \left(p(t)^+ \cdot (1+\epsilon)^{1-y}\right) \times \sum_{u_i \in \mathcal{U}(t)} d_i(p^*(t)) \\
&\le (1+\epsilon) \times \left(p(t)^+ \cdot (1+\epsilon)^{-y}\right) \\
&\quad \times \sum_{u_i \in \mathcal{U}(t)} d_i\left(p(t)^+ \cdot (1+\epsilon)^{-y}\right) \\
&\le (1+\epsilon) \times \hat{\pi}
\end{aligned} \tag{15}$$

Algorithm 2: Price-Incentive Resource Auction Algorithm (PIRA)

Input: Service bids of users $\mathcal{U}(t)$ in the time slot t.
Output: Resource price $p(t)$;
Resource allocation scheme $A(t)$.

```
// Phase 1
Initialize i'' ← N(t) − 1;
for each i = {1, 2, ..., N(t)} do
    Pick out the user u_[i] ranked at the i-th place in the ascending-ordered Θ;
    if p^+_[i] ≥ c̃ · p^e(t) · (1 + γ)/r then
        i'' ← i;
        break;
    else
        Reject the service bid of user u_[i];
// Phase 2
for each i = {i'', i'' + 1, ..., N(t)} do
    p(t)^+ ← p^+_[i];
    if i = i'' then
        p(t)^− ← c̃ · p^e(t) · (1 + γ)/r;
    else if i ≥ i'' + 1 then
        p(t)^− ← p^+_[i−1] + σ;
    p(t), A(t) ← RORP( p(t)^−, p(t)^+ );
    if p(t) =null and A(t) =null then
        Reject the service bid of user u_[i];
    else
        Accept the service bid of users Ũ(t) = { u_[j] }_{j=i}^{N(t)};
        return p(t) and A(t) ;
return null;
```

where the inequality (15) arises from the fact that the user resource demand is non-increasing with the unit cloud resource price. Henceforth, the approximation ratio of our RORP algorithm is proved to be $(1+\epsilon)$.

In the RORP algorithm, the revenue of *X* candidate prices needs to be respectively calculated by means of simulating each user's rational cloud resource purchase. Under an arbitrary candidate price $\hat{p}_x$, it takes the time of *O*(1) to simulate *N*(*t*) users conducting the rational cloud resource purchase in parallel. To summarize, the parallel computation complexity of the RORP algorithm is *O*(*X*). ■

Generally speaking, we novelly convert the non-convex optimization problem with a continuous decision range into the one with a discrete decision domain, and thus devise a computational-efficient approximate algorithm called RORP to acquire a near-optimal result. As indicated in Theorem 1, the constant parameter ϵ strikes a trade-off between computational efficiency and approximation ratio, as exemplified in Table II. With the RORP algorithm, the final spot unit resource price *p*(*t*) together with the resource allocation scheme *A*(*t*) is derived in Phase 2.

From the above, our price-incentive resource auction mechanism proceeds in the two phases which are Phase 1 and Phase 2. We encapsulate the details of this two phases in the Price-Incentive Resource Auction algorithm namely PIRA, as shown in Algorithm 2. The PIRA algorithm is executed at each time slot *t*.

Overall Computation Complexity Analysis: The PIRA algorithm comprises Phase 1 and Phase 2. In Phase 1, we assume that $N_1(t)$ users are excluded from the cloud user subset $\mathcal{U}(t)$, where $N_1(t) \leq N(t)$. When deciding whether to eliminate a user's service bid, we need to justify whether the corresponding $p^+_{[i]}$ is no less than $\lfloor p \rfloor$. Therefore, the computation complexity of the Phase 1 is $O(N_1(t))$. In Phase 2, we suppose that $\mathcal{X}$ discrete candidate prices are totally put into trial before the final spot unit resource price *p*(*t*) is determined. At each temporary price setting, we simulate each user's rational cloud resource purchase in parallel, takeing the time of *O*(*1*). Thus, the computation complexity is $O(\mathcal{X})$ in Phase 2. To sum up, the computation complexity is $O(N_1(t) + \mathcal{X})$ for the PIRA algorithm.

E. Properties of Resource Auction Mechanism

Theorem 2 (Budget Balance): The user's bidding budget b_i can always cover the service payment charged for the amount $a_i(t)$ of purchased cloud resources.

Proof: Given the space limitation, the detailed proof for Theorem 2 is presented in Appendix B of the supplementary material. ■

Theorem 3 (Truthfulness): The user has no incentive to misreport her bidding budget.

Proof: Given the space limitation, the detailed proof for Theorem 3 is presented in Appendix C of the supplementary material. ■

Theorem 4 (Envy-Freeness): The user always prefers her own purchased amount of resources to that of others.

Proof: Given the space limitation, the detailed proof for Theorem 4 is presented in Appendix D of the supplementary material. ■

V. Evaluation

A. Experimental Setup

Performance Data: Since the AI/ML jobs are known for their high computation intensity [50], [51], thus we choose eight representative AI/ML-related applications provided in Spark MLlib [52] as the service types s_i proposed by service requests, which are Classification, Naive.Bayes, Regression, KMeans, Spearman, PCA, ALS and Summary.Stats. A recent performance prediction framework called Ernest [12] proposed a general job performance model for a variety of computation-intensive applications including these eight service types. Through parametric regression analysis, the relationship between the job runtime and the number of allocated resources is characterized as (1), where the regression coefficients θ_{s_i}, ϕ_{s_i}, η_{s_i} and ω_{s_i} of our eight service types are explicitly given in [12]. Therefore, in our experiments, a service request is proposed with the service type which is randomly drawn

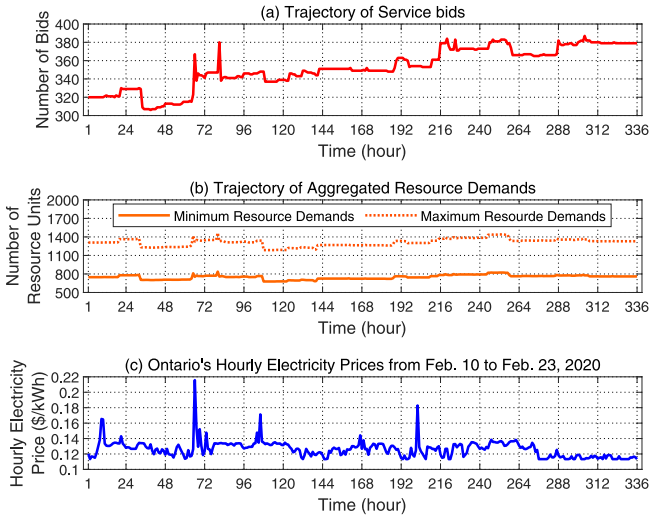


Fig. 3. Overview of Trace Data.

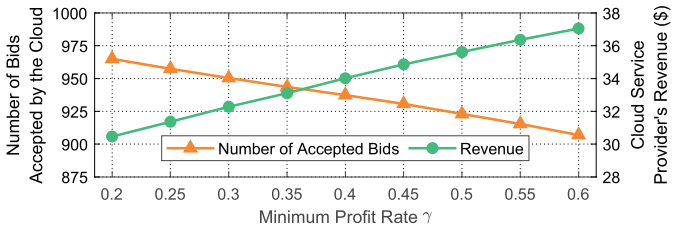


Fig. 4. Number of Accepted Bids, and Cloud Service Provider's Revenue under Different Minimum Profit Rates γ.

from the Classification, Naive.Bayes, Regression, KMeans, Spearman, PCA, ALS and Summary.Stats.

Workload Data: For the trace-driven experiments, we extract the trajectory of service requests from the public Microsoft Azure Cluster trace dataset [53]. The public Azure dataset documents the VM running status in one of Azure's geographical regions during one month from November 16, 2016 to February 16, 2017, in which the activities of VM creation and deletion are recorded across time. In the Azure dataset, the subscription is a natural unit representing an Azure subscriber who performs logically-related jobs. The Azure subscriber can create several VMs based on their demands across time, where the number of vCPU cores per VM can be designated by the Azure subscriber.

We conceive each of VM created by the Azure subscriber as a single service request, and the VM lifetime is regarded as the temporal interval $[t_i^-, t_i^+]$ subscribed by the service request. When the VM created by the Azure subscriber is active during the time slot, then a service bid would be correspondingly placed at this time slot. Additionally, the minimum resource demand a_i^- of a service request is set based on the number of vCPU cores allocated to the VM, while the maximum resource demand a_i^+ is scaled as $1.75\times a_i^-$. The corresponding QoS requirement $[T_i^-, T_i^+]$ of the service request is mapped from $[a_i^-, a_i^+]$, based on (1).

The duration τ of each time slot is set as one hour, in accordance with the minimum billing cycle of Microsoft Azure [54]. To simplify the experiment, we choose 336 time slots (i.e., 14 days) of data from the Azure workload trace, and randomly select a fraction of 100 Azure subscribers from the entire dataset. Fig. 3(a) shows the number of service bids placed by these 100 Azure subscribers across time slots, where there are approximately 300 to 400 service bids concurrently placed in a time slot. Meanwhile, Fig. 3(b) demonstrates the trajectory of aggregated resource demands from our selected 100 Azure subscribers.

Electricity Price Data: We collect the real-world trace data for electricity price from the Independent Electricity System Operator (IESO) [55], who manages the Ontario's power system. The Ontario's hourly electricity price is updated by IESO per hour, and the electrical customers should pay for their energy costs according to the instant hourly electricity price. For the trace-driven experiments, we adopt the Ontario's hourly electricity prices from February 10 to February 23, 2020 for 14 days, aligned with our selected Azure workload data. The trace of Ontario's hourly electricity prices is illustrated in Fig. 3(c). Note that, the average value over the trace data for Ontario's hourly electricity prices is 0.1262 \$/*kWh*, with the standard deviation of 0.0106 \$/*kWh*. At each run of our numerical experiments, the electricity price (in unit of \$/kWh) is randomly drawn from a normal distribution as $\mathcal{N}(0.1262,\, 0.0106^2)$.

Parameter Settings: For the cloud service provider, each physical machine of cloud has a resource capacity r of 10. Each physical machine is assumed to expend an average of 500 Watts based on [56]. Hence, the average energy cost $\widetilde{c}$ to operate a physical machines is estimated as $1.2 * 500 = 600$ Watts, where the power effectiveness usage PUE is considered as 1.2 according to [57]. Meanwhile, the minimum profit rate γ required by the cloud service provider is configured as 0.2 (unless specified). With regard to the bidding budget, we follow the assumption adopted in [8], [39] to generate the bidding budget b_i (in unit of \$) for each service request according to a normal distribution of $((a_i^- + a_i^+)/2) \cdot \mathcal{N}(0.02, 0.015^2)$. The approximation ratio $(1+\epsilon)$ is set as 1.02 in the RORP algorithm.

B. Numerical Experiments

To clearly demonstrate the experimental result in numerical experiments, we simply perform our PIRA algorithm for one time slot here. Each of the following experimental results is averaged over several runs to reflect the randomized nature of auction.

1) Impact of Minimum Profit Rate γ: We firstly study the impact of minimum profit rate γ on the auction outcome, i.e., the number of accepted bids and the cloud service provider's revenue. Here, we set 1,000 users concurrently placing service bids to the cloud in a time slot. Fig. 4 demonstrates the auction outcome under different minimum-profit-rate settings, where each of experimental results is averaged over 200 runs. It can been observed that, less service bids are accepted by the cloud when γ is set higher. This is because more service bids with relatively limited budgets are rejected. The rejected service bids cannot afford to satisfy such a high profitability objective γ. But in the meantime, the revenue earned by the

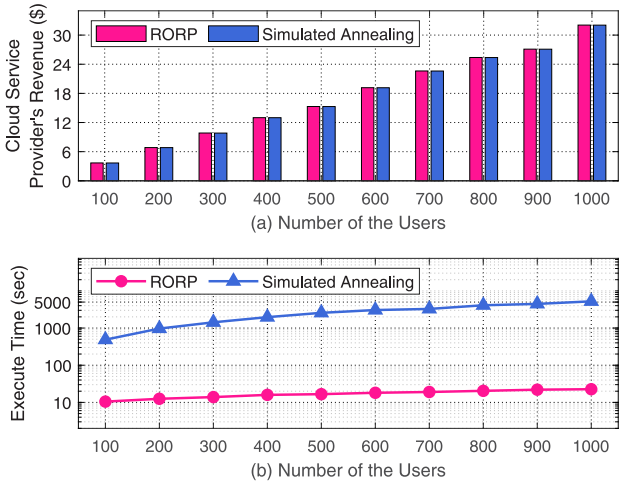


Fig. 5. Performance Comparison between our RORP Algorithm, and the Simulated Annealing Algorithm [58].

cloud service provider increases with a higher γ configured. It happens because the cloud service provider sells more cloud resources to bidders with the higher budget, for the sake of its higher profitability objective γ. The above experimental analysis matches with our expectation that γ should balance the interests of users and the cloud service provider.

2) Evaluation of RORP Algorithm Performance: Our RORP algorithm plays an important part in our resource auction mechanism. With the maximum cloud user subset determined in the PIRA algorithm, our RORP algorithm is responsible for finding out an optimal resource price which gains an approximate maximum revenue. Given its importance, we intend to evaluate the effectiveness and efficiency of our RORP algorithm. In specific, we conduct the comparative experiment with a state-of-the-art algorithm named *Simulated Annealing* [58]. As a renowned nonlinear optimization approach, the *Simulated Annealing* algorithm can approximate the optimal price of unit cloud resource where the highest revenue is earned. Note that, our comparative experiment is performed on Windows 10 where the processor is Intel Core i7-5500U 2.4GHz with the memory size of 12 GB.

Fig. 5 shows the comparative results between our RORP algorithm and the *Simulated Annealing* algorithm. Note that, each experimental result about our RORP algorithm is taken an average over 200 runs, while that about the *Simulated Annealing* algorithm is averaged over 10 runs. In terms of algorithmic effectiveness, we compare these two competitive algorithms on the cloud service provider's revenue. Under varied scales of bidding users, the difference on the gained revenue between RORP and *Simulated Annealing* is extremely small. Nevertheless, from the aspect of algorithmic efficiency, a significant quantitative difference on the execute time exists between these two approaches. Here, the execute time implying the computation cost is measured, where each user's resource demand is sequentially calculated under a certain resource price. Our RORP algorithm's execute time is far less than that of the *Simulated Annealing* algorithm, while our RORP algorithm gains the revenue with little difference to the *Simulated Annealing* algorithm. It implies the advancement of our RORP algorithm. Also, our RORP algorithm is expected to gain much less execute time when each user's resource demand is resolved in parallel.

3) Auction Property Validation: We further conduct numerical analysis to experimentally verify the properties of budget balance and envy-freeness for our price-incentive resource auction mechanism. Similar experimental results can be obtained in the subsequent trace-based experiment as well. Here, each experimental result is averaged over 200 runs.

Budget Balance: We set 100 users who concurrently place service bids to the cloud service provider in a time slot. The service payment determined by the PIRA algorithm and the bidding budget are compared across different users in Fig. 7. All these 100 users' bidding budgets respectively cover the service payment of their own, where a user averagely consumes 64.71% of her own bidding budget for service payment. Given this, the property of budget balance is verified through experiments.

Envy-Freeness: To better demonstrate the numerical result which satisfies the property of envy-freeness, we simplify the experimental scenario with eight users $u_1 \sim u_8$ concurrently placing service bids. Each user u_i would be respectively allocated a_i units of cloud resources through our PIRA algorithm, if her service bid is accepted. The user u_i gains the net-utility $v_i(a_i)$ on her own allocated resources a_i, with her QoS requirement fulfilled and her bidding budget balanced. For each user u_i, the net-utility $v_i(a_i)$ on her own allocated resources a_i is compared with the net-utility $v_i(a_j)$ on other users's allocated resources a_j where $j \neq i$, as shown in Fig. 6. It can be seen that, the net-utility $v_i(a_i)$ on her own purchased resources a_i is always the greatest, no matter for the user $u_1 \sim u_8$, which embodies the property of envy-freeness during resource auction.

C. Simulation Experiments Using Real Trace Data

Based on the real trace data, we then study the effectiveness of our price-incentive resource mechanism. In specific, we investigate the performance of our proposed PIRA algorithm respectively from the both sides of cloud service provider and users. For the cloud service provider, we testify whether the minimum profit rate γ can be ensured by the PIRA algorithm over time slots. For users, we evaluate the service capability of our PIRA algorithm, measured by the number of the accepted service bids. A state-of-the-art solution called *Revenue-Max* is compared with our PIRA algorithm. The *Revenue-Max* algorithm is an unilateral solution which only intends to gain the maximum revenue for the cloud service provider regardless of the user interests. The objective of revenue maximization in the *Revenue-Max* algorithm is achieved by means of the same approximate technique adopted in our proposed RORP algorithm.

Fig. 8(a) shows the cloud service provider's revenue earned under our PIRA algorithm across time slots, comparing with the *Revenue-Max* algorithm and the baseline of minimum profit-rate guarantee. The minimum profit rate γ is preconfigured by the cloud service provider according to its profitability goal. It can be seen that the revenue baseline corresponding to the minimum profit-rate guarantee fluctuates

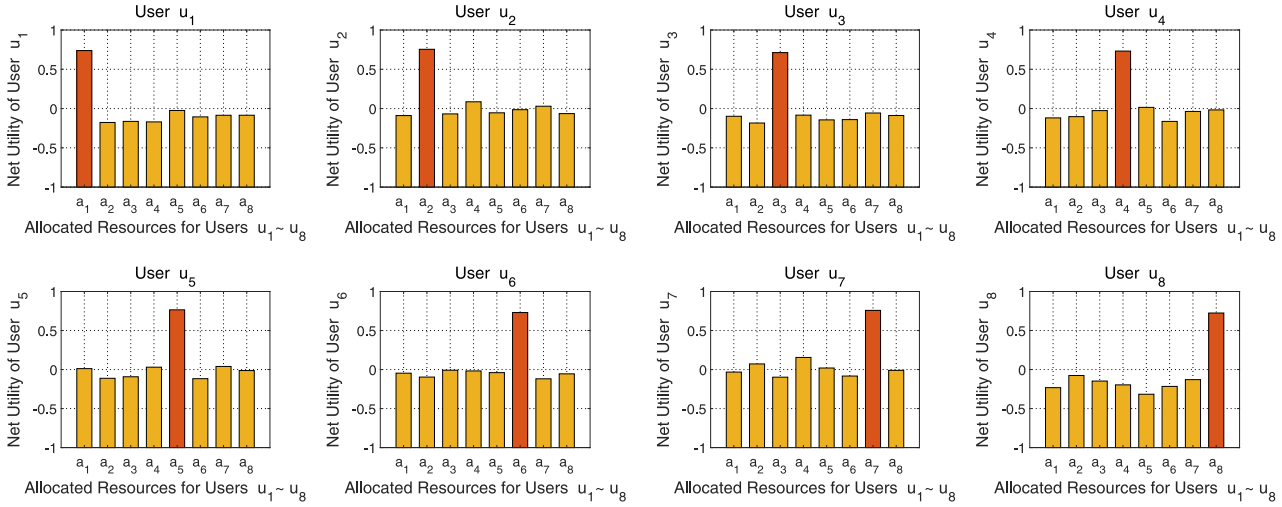


Fig. 6. Comparison of Net Utility on Various Resource Allocations across Different Users.

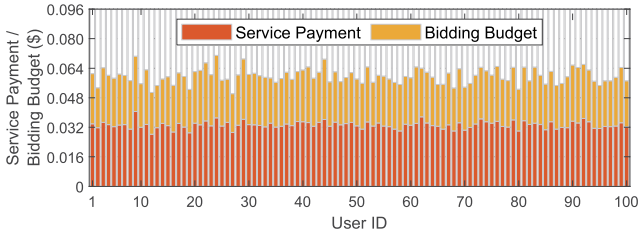


Fig. 7. Service Payment v.s. Bidding Budget across Different Users.

with the hourly-updated electricity price. The cloud service provider's revenue under our PIRA algorithm is always above the minimum profit-rate baseline, which meets our expectation for our resource auction mechanism that at least γ profit rate should be guaranteed. Additionally, the cloud service provider's revenue under our PIRA algorithm fulfills the minimum profit-rate requirement exactly over the baseline. As long as the minimum profit rate γ is fulfilled, the cloud service provider would like to surrender part of the revenue relative to that in the *Revenue-Max* algorithm by publishing a competitively low price, with the aim of stimulating cloud users as many as possible.

Fig. 8(b) shows the number of service bids accepted by the cloud service provider under different algorithms. Differing from our PIRA algorithm balancing the interests between users and the cloud service provider, the state-of-the-art *Revenue-Max* algorithm unilaterally targets to gain the maximum revenue for the cloud service provider regardless of the user interests. It can be obtained from Fig. 8(b) that our proposed PIRA algorithm accepts more service bids than the *Revenue-Max* algorithm, therefore capturing a higher fraction of cloud users in the marketplace. The comparative result coincides with our envision for the PIRA algorithm incentivizing the maximum users willing to be served at cloud.

Furthermore, we also make detailed comparative analysis between the approaches of PIRA and *Revenue-Max*. Since the *Revenue-Max* algorithm neglects the user interests but merely focuses on gaining the maximum revenue, thus the corresponding spot unit resource price $p(t)$ is usually set higher than the PIRA algorithm across time slots, as shown in Fig. 8(c). However, thanks to a lower, more competitive spot unit resource price $p(t)$ published by the PIRA algorithm, our price-incentive resource auction mechanism can attract more cloud users. Accordingly, our PIRA algorithm requires more active physical machines serving for cloud users, as shown in Fig. 8(d). With our PIRA algorithm, the cloud service provider can incentivize more cloud users, notwithstanding an acceptable sacrifice of revenue in comparison with the *Revenue-Max* approach.

VI. Conclusion

In this article, we study the market-based cloud pricing strategy and put forward a price-incentive resource auction mechanism. The proposed mechanism balances the interests between users and the cloud service provider, specifically stimulating the maximum cloud users while a minimum profit rate is ensured for the cloud service provider. Facing the challenges of (1) the NP-hardness of finding the optimal solution, (2) the segmentation and concavity of the net-utility function, and (3) the correlation between decisions on resource pricing and allocation, we adopt the decomposition approach and the approximate optimization technique, and design a computational-efficient resource auction algorithm called PIRA which dynamically determines the resource price and allocation over time slots. Our proposed mechanism is theoretically proved to be budget balanced, truthfulness, and envy-free. The efficacy of our proposed resource auction mechanism is validated by simulation experiments based on the real-world dataset. Our research work is expected to have potential contribution towards the resource auction mechanism design in the cloud marketplace.

Moreover, there are also several avenues for our future work. Firstly, a more fine-grained cloud resource allocation scheme should be customized for different kinds of jobs. It further involves a more exquisite management on varied cloud resources (e.g., CPU units, memory size, disk storage, and network bandwidth). In computation-intensive jobs, the bottleneck resource is generally the CPU unit. But in network-intensive jobs, the bottleneck resource turns to be the network

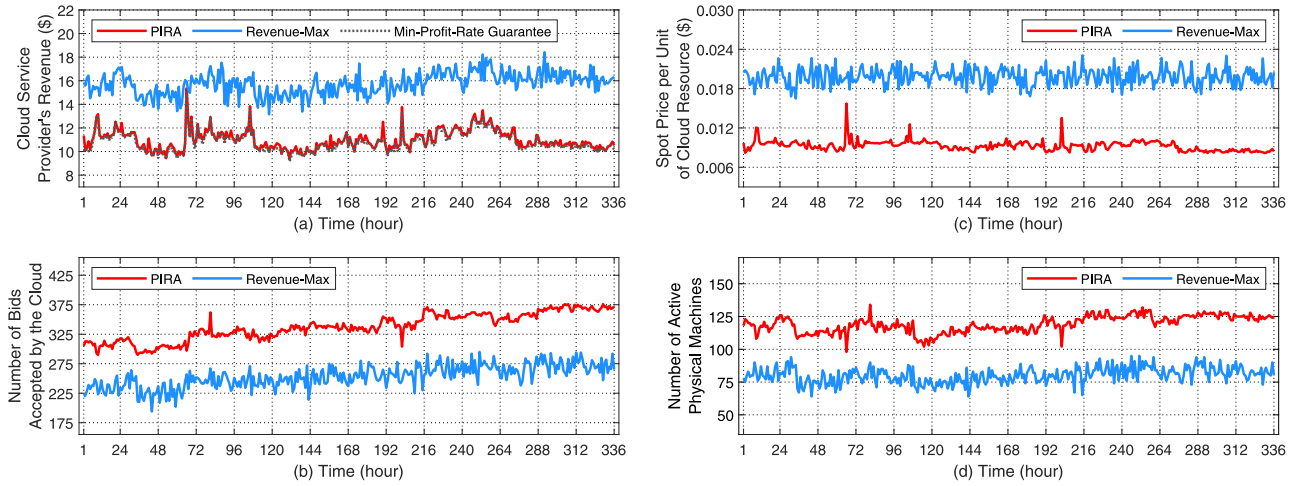


Fig. 8. Comparative Simulation Results based on Real Trace Data.

bandwidth. Thus, for different kinds of jobs, a tailored resource allocation strategy should be deployed. Secondly, it is of great value to verify the cloud resource auction mechanism in a real-world environment. The evaluation results obtained from an in-the-wild experiment have more credibility, although there is much difficulty to construct a realistic experiment environment where numerous real bidders participate in the cloud resource auction. The realistic experiment may provide more insights on the resource auction mechanism design in the real-world cloud environment.

REFERENCES

[1] *Amazon Elastic Compute Cloud (Amazon EC2)*. Accessed: Apr. 29, 2020. [Online]. Available: https://aws.amazon.com/ec2/

[2] *Google Cloud Computing Services*. Accessed: Apr. 29, 2020. [Online]. Available: https://cloud.google.com

[3] *Microsoft Azure Cloud Computing Services*. Accessed: Apr. 29, 2020. [Online]. Available: https://azure.microsoft.com/en-us

[4] C. Wu, R. Buyya, and K. Ramamohanarao, "Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 1–36, 2019.

[5] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online auctions in IaaS clouds: Welfare and profit maximization with server costs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1034–1047, Apr. 2017.

[6] S. Hou, W. Ni, S. Zhao, B. Cheng, S. Chen, and J. Chen, "Decentralized real-time optimization of voltage reconfigurable cloud computing data center," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 577–592, Jun. 2020.

[7] I. A. Kash and P. B. Key, "Pricing the cloud," *IEEE Internet Comput.*, vol. 20, no. 1, pp. 36–43, Jan./Feb. 2016.

[8] B. Zheng, L. Pan, S. Liu, and L. Wang, "An online mechanism for purchasing IaaS instances and scheduling pleasingly parallel jobs in cloud computing environments," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 35–45.

[9] W. Borjigin, K. Ota, and M. Dong, "In broker we trust: A double-auction approach for resource allocation in NFV markets," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1322–1333, Dec. 2018.

[10] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry, "On profit-maximizing envy-free pricing," in *Proc. Annu. ACM-SIAM Symp. Discr. Algorithms (SODA)*, 2005, pp. 1164–1173.

[11] D. K. Foley, "Resource allocation and the public sector," *Yale Econ. Essays*, vol. 7, no. 1, pp. 45–98, 1967.

[12] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 363–378.

[13] *Amazon EC2 Spot Instances*. Accessed: Apr. 29, 2020. [Online]. Available: https://aws.amazon.com/ec2/

[14] S. Yang, L. Pan, and S. Liu, "An online algorithm for selling your reserved IaaS instances in Amazon EC2 marketplace," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2019, pp. 296–303.

[15] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, "Learning cloud dynamics to optimize spot instance bidding strategies," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 2762–2770.

[16] N. Kamiyama, "Virtual machine trading in public clouds," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 403–415, Mar. 2020.

[17] J. Wan, R. Zhang, X. Gui, and B. Xu, "Reactive pricing: An adaptive pricing policy for cloud providers to maximize profit," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 941–953, Dec. 2016.

[18] P. Cong *et al.*, "Developing user perceived value based pricing models for cloud markets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 12, pp. 2742–2756, Dec. 2018.

[19] Z. Zheng, R. Srikant, and G. Chen, "Pricing for revenue maximization in inter-datacenter networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 28–36.

[20] C. Wang, B. Urgaonkar, G. Kesidis, A. Gupta, L. Y. Chen, and R. Birke, "Effective capacity modulation as an explicit control knob for public cloud profitability," *ACM Trans. Auton. Adapt. Syst.*, vol. 13, no. 1, pp. 1–25, 2018.

[21] A. Motamedi, H. Zareipour, and W. D. Rosehart, "Electricity price and demand forecasting in smart grids," *IEEE Trans. Smart Grid*, vol. 3, no. 2, pp. 664–674, Jun. 2012.

[22] C. Woo *et al.*, "Electricity price behavior and carbon trading: New evidence from California," *Appl. Energy*, vol. 204, pp. 531–543, Oct. 2017.

[23] M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis, and E. Agiatzidou, "Energy-aware cost prediction and pricing of virtual machines in cloud computing environments," *Future Gener. Comput. Syst.*, vol. 93, pp. 442–459, Apr. 2019.

[24] N. Nasiriani, C. Wang, G. Kesidis, B. Urgaonkar, L. Y. Chen, and R. Birke, "On fair attribution of costs under peak-based pricing to cloud tenants," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 1, pp. 1–28, 2016.

[25] A. Sarker, Z. Li, W. Kolodzey, and H. Shen, "Opportunistic energy sharing between power grid and electric vehicles: A game theory-based pricing policy," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 1197–1207.

[26] C. Qiu, H. Shen, and L. Chen, "Towards green cloud computing: Demand allocation and pricing policies for cloud service brokerage," *IEEE Trans. Big Data*, vol. 5, no. 2, pp. 238–251, Jun. 2019.

[27] J. Huang, J. Zou, and C.-C. Xing, "Competitions among service providers in cloud computing: A new economic model," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 866–877, Jun. 2018.

[28] R. Landa, M. Charalambides, R. G. Clegg, D. Griffin, and M. Rio, "Self-tuning service provisioning for decentralized cloud applications," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 197–211, Jun. 2016.

[29] C. Jiang, Y. Chen, Q. Wang, and K. J. R. Liu, "Data-driven auction mechanism design in IaaS cloud computing," *IEEE Trans. Services Comput.*, vol. 11, no. 5, pp. 743–756, Sep./Oct. 2018.
[30] H. Sun, H. Yu, and G. Fan, "Contract-based resource sharing for time effective task scheduling in fog-cloud environment," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 1040–1053, Jun. 2020.
[31] L. Lu, J. Yu, Y. Zhu, and M. Li, "A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 720–733, Apr. 2018.
[32] Y. Zhang, A. Ghosh, V. Aggarwal, and T. Lan, "Tiered cloud storage via two-stage, latency-aware bidding," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 176–191, Mar. 2019.
[33] S. Hosseinalipour and H. Dai, "A two-stage auction mechanism for cloud resource allocation," *IEEE Trans. Cloud Comput.*, early access, Feb. 26, 2019, doi: 10.1109/TCC.2019.2901785.
[34] V. Arabnejad, K. Bubendorfer, and B. Ng, "Budget and deadline aware e-Science workflow scheduling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 29–44, Jan. 2019.
[35] Z. Feng, O. Schrijvers, and E. Sodomka, "Online learning for measuring incentive compatibility in ad auctions?" in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2729–2735.
[36] D. Zhang *et al.*, "Near-optimal and truthful online auction for computation offloading in green edge-computing systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 880–893, Apr. 2020.
[37] Y. Zhu, S. D. Fu, J. Liu, and Y. Cui, "Truthful online auction toward maximized instance utilization in the cloud," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2132–2145, Oct. 2018.
[38] G. Baranwal and D. P. Vidyarthi, "A truthful and fair multi-attribute combinatorial reverse auction for resource procurement in cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 6, pp. 851–864, Nov./Dec. 2019.
[39] B. Yang, Z. Li, S. Jiang, and K. Li, "Envy-free auction mechanism for VM pricing and allocation in clouds," *Future Gener. Comput. Syst.*, vol. 86, pp. 680–693, Sep. 2018.
[40] A. Prabhakaran and L. J., "Cost-benefit analysis of public clouds for offloading in-house HPC jobs," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, 2018, pp. 57–64.
[41] C. Chen, W. Wang, and B. Li, "Performance-aware fair scheduling: Exploiting demand elasticity of data analytics jobs," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 504–512.
[42] *IONOS Enterprise Cloud*. Accessed: Aug. 23, 2020. [Online]. Available: https://www.ionos.com/enterprise-cloud
[43] *CloudSigma*. Accessed: Aug. 23, 2020. [Online]. Available: https://www.cloudsigma.com
[44] C. Chen, W. Wang, and B. Li, "Speculative slot reservation: Enforcing service isolation for dependent data-parallel computations," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 549–559.
[45] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An SMDP-based service model for interdomain resource allocation in mobile cloud networks," *IEEE Trans. Veh. Technol.*, vol. 61, no. 5, pp. 2222–2232, Jun. 2012.
[46] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li, "Stackelberg game approach for energy-aware resource allocation in data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3646–3658, Dec. 2016.
[47] L. A. Barroso, U. Hlzle, and P. Ranganathan, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 3rd ed. London, U.K.: Morgan & Claypool, 2018.
[48] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[49] J. F. Bonnans, J. C. Gilbert, C. Lemarechal, and C. A. Sagastizabal, *Numerical Optimization: Theoretical and Practical Aspects (Universitext)*. New York, NY, USA: Springer-Verlag, 2006.
[50] I. Stoica *et al.* (2017). *A Berkeley View of Systems Challenges for AI*. [Online]. Available: https://arxiv.org/abs/1712.05855
[51] D. E. Womble, M. Shankar, W. Joubert, J. T. Johnston, J. C. Wells, and J. A. Nichols, "Early experiences on summit: Data analytics and AI applications," *IBM J. Res. Develop.*, vol. 63, no. 6, pp. 1–9, 2019.
[52] *Spark MLlib*. Accessed: Aug. 23, 2020. [Online]. Available: https://spark.apache.org/mllib/
[53] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proc. 26th ACM Symp. Oper. Syst. Principles (SOSP)*, 2017, pp. 153–167.
[54] *Pricing Calculator—Configure and Estimate the Costs for Azure Products*. Accessed: Apr. 29, 2020. [Online]. Available: https://azure.microsoft.com/en-us/pricing/calculator/
[55] *Electricity Pricing in Ontario*. Accessed: Apr. 29, 2020. [Online]. Available: http://www.ieso.ca/power-data
[56] C. Wang *et al.*, "Recouping energy costs from cloud tenants: Tenant demand response aware pricing design," in *Proc. ACM 6th Int. Conf. Future Energy Syst. (e-Energy)*, 2015, pp. 141–150.
[57] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2648–2658, Oct. 2014.
[58] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

Songyuan Li received the B.Eng. degree in computer science and technology from the Beijing University of Posts and Telecommunications in 2018. He is currently pursuing the master's degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunication, China. He was a recipient of China National Scholarship in 2019. He has published articles in international journals and conference proceedings, including the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, the *Peer-to-Peer Networking and Applications*, the *International Journal of Web and Grid Services*, IEEE ICWS, IEEE SCC, and IEEE ISPA. His current research interests include cloud computing, edge computing, services computing, performance evaluation, and optimization.

Jiwei Huang (Member, IEEE) received the B.Eng. and Ph.D. degrees in computer science and technology from Tsinghua University, in 2009 and 2014, respectively. He was a Visiting Scholar with the Georgia Institute of Technology. He is currently a Professor and the Dean with the Department of Computer Science and Technology, China University of Petroleum, Beijing, China, and the Director with the Beijing Key Laboratory of Petroleum Data Mining. He has published one book and more than 50 articles in international journals and conference proceedings, including the IEEE TRANSACTIONS ON SERVICES COMPUTING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, ACM SIGMETRICS, IEEE ICWS, and IEEE SCC. His research interests include services computing, cloud computing, and performance evaluation. He is a Member of the ACM.

Bo Cheng (Member, IEEE) received the Ph.D. degree in computer science and engineering from the University of Electronic Science and Technology of China in 2006. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. His current research interests include network services and intelligence, Internet of Things technology, communication software, and distributed computing. He serves on the editorial board of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He is a Member of the ACM.