Future Generation Computer Systems 86 (2018) 680–693

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Envy-free auction mechanism for VM pricing and allocation in clouds

Bo Yang [a,b,c], Zhiyong Li [a,b,*], Shilong Jiang [d], Keqin Li [a,e]

[a] College of Information Science and Engineering, Hunan University, Changsha, 410008, China
[b] National Supercomputing Center in Changsha, Changsha, 410008, China
[c] College of Information and Management, Hunan University of Finance and Economics, Changsha, 410205, China
[d] PKU-HKUST Shenzhen-HongKong Institution, Shenzhen, 518000, China
[e] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

HIGHLIGHTS

- Multiple criteria decision making technique is employed for measuring VMs.
- Envy-freeness, truthfulness and near optimal revenue can be achieved.
- Proposed mechanism can provide computationally efficient solutions.
- Especially being suitable for large-scale and over-supplied cloud markets.

ARTICLE INFO

Article history:
Received 31 December 2017
Received in revised form 19 April 2018
Accepted 19 April 2018
Available online 1 May 2018

Keywords:
Auction mechanism
Cloud computing
Envy-freeness
Truthfulness
VM allocation and pricing

ABSTRACT

A major challenging problem in clouds is designing efficient mechanisms for virtual machine (VM) allocation and pricing. Failure to fully consider the incentives of cloud providers and customers can cause undesirable outcomes, such as no envy-freeness and untruthfulness, which may lead to system instability and relatively low profit for cloud providers. In this study, we proposed a combinatorial auction-based mechanism to address such problem in the presence of multiple types of VMs in a single provider scenario. The proposed mechanism combines two general ideas: *consensus estimate* that can avoid market manipulation and yields an approximate optimal target revenue with the consensus estimate technology, and *RevenueExtraction* that can determine the winners and equally shares the target revenue generated by *consensus estimate* among them with a single sale price. Using the two ideas, the proposed mechanism can simultaneously promise truthfulness and envy-freeness while achieving an approximate optimal revenue. The results of extensive simulation experiments demonstrate that our schemes can efficiently deliver stable and desirable performance, especially in large-scale and over-supplied cloud markets.

1. Introduction

1.1. Motivation

Cloud computing provides an attractive paradigm for offering computing services in the pay-as-you-go model. Usually, the provided services refer to Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) [1]. To deliver satisfying quality of service and efficient resource allocation, simultaneously meeting the economic incentives of cloud users and cloud providers (CPs) has become a fundamental and unavoidable problem in cloud computing. Compared with the fixed-priced mechanism, auction-based mechanisms can provide more incentives to users to adjust consumption patterns according to demand and supply, and also bring higher profit for Cps [2]. For example, Amazons EC2 [3] has successfully adopted an auction-like approach to expand its pricing plans with Spot instances. Consequently, employing auction mechanisms to make decisions for resource allocation and pricing in cloud markets has attracted increasing interests from both the research community and the industry [4].

In practice, it is undoubted that CPs always pursue the maximization of revenue. Therefore, they expect everyone participating in auctions to report true bids such that the maximization of their benefits can be guaranteed, that is, the auctions employed by them are *truthful*. However, most truthful auctions, e.g., the Vickrey–Clarke–Groves (VCG) auction and the ascending auction,

* Corresponding author at: College of Information Science and Engineering, Hunan University, Changsha, 410008, China.
E-mail addresses: bo_yang@hnu.edu.cn (B. Yang), zhiyong.li@hnu.edu.cn (Z. Li), jiangshilong03@126.com (S. Jiang), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.future.2018.04.055
0167-739X/

for selling multiple goods to bidders can discriminate between bidders by selling identical goods at different prices [5]. Likely, in cloud environment, a cloud customer usually requests a bundle of distinct type commodities (such as VMs or containers) from cloud providers [6,7], so price discrimination also exists in cloud markets while such auctions are conducted. Price discrimination is problematic as pointed out by economic literature [8], which makes a number of users envy the results of other users, placing the system in an unstable state, and in several cases also forbidden by international commerce law [9]. *Envy-free* allocations were suggested tackling such problem by classical literatures [5,8]. The key property of such allocations is that no one envies the allocation and the price charged to anyone else. Obviously, envy-freeness promising a fair treatment and truthfulness bringing highly expected revenue are the most important properties of auction-based mechanisms.

However, it is hard to simultaneously achieve both envy-freeness and truthfulness in the auction for selling multiple goods to bidders in cloud markets. To tackle the problem, several compromising scenarios are acceptable, such as, under the premise of ensuring envy-freeness, the auction-based mechanism can provide either truthfulness with high probability or suboptimal economic efficiency while promising truthfulness. Hence, a trade-off frequently exists among envy-freeness, truthfulness and economic efficiency.

In addition, clouds present such a promise of providing infinite capacity of resources, and recent studies also report that the overall utilization in large data centers is lower than 50% most of the time [10,11]. That is, the resource provisioning of CPs is often oversupplied. However, traditional truthful mechanisms (such as the VCG mechanism) usually fail to work in this scenario due to they are born to serve for undersupplied markets. The price produced by them in this case is very low such that CPs have little profit, because of having to accept the bidders with low bid.

Furthermore, due to the lack of effective estimate method for different type VMs, the provisioning VMs often are regarded as type-oblivious commodities to respond to the heterogeneity of VMs [12,13]. That is, either a single type of VMs exists in the cloud market, or VMs are substitutes in which a high-end VM is equivalent to a number of low-end VMs. While designing auction-based approaches for selling multiple type VMs to users in cloud markets, such estimates are inaccurate and unfair for users and cannot be adapted straightforwardly to handle the existing VM heterogeneity.

1.2. Contributions

In this paper, we model the resource allocation problem in a cloud system as a combinatorial auction. The service provider is defined as a seller who has different type VMs to be leased or sold to cloud users, whereas the cloud users are defined as buyers. The buyer asks a bundle of VMs with different configurations from the seller. We solve the problem of VMs allocation and pricing in a single provider scenario, aiming to achieve the maximization of cloud providers' revenue, envy-freeness and truthfulness. Our main contributions are summarized as follows.

The sold items in our schemes are extended to multiple types of resources, rather than type-oblivious commodities, to be consistent with the diversity of VMs provisioning in cloud computing. Moreover, the multiple criteria decision making technique is adopted in order to fairly measure the bundles of VMs requested by distinct users with different requirements based on the provider's criteria.

A combinatorial auction mechanism framework (*EFM*) is presented to fulfill our goals. It combines two general ideas: *consensus estimate* that can avoid market manipulation and yields an approximate optimal target revenue with the consensus estimate technology, and *RevenueExtraction* that equally shares the target revenue generated by *consensus estimate* among winners with a single sale price. Using the two ideas, the proposed mechanism can promise the two properties of truthfulness and envy-freeness while achieving an approximate optimal revenue. It can provide computationally efficient solutions being suitable for execution in short time window auctions, and deliver stable and desirable performance, especially in large-scale and over-supplied cloud markets.

The rest of this paper is organized as follows. We first discuss related works in Section 2, and then introduce the system model in Section 3. We provide the formulation of the resources allocation problem and present the detailed description of the proposed algorithms in Sections 4 and 5, respectively. We discuss the simulation results that demonstrate the effectiveness of our approach in Section 6. Finally, the conclusions are drawn in Section 7.

2. Related works

Development of efficient and effective mechanisms to decide computing resources allocating, pricing and provisioning has attained increasing interest. Existing investigations have approached this problem from various points of view, such as conventional models, economic and game-theoretic models [14–16,11,17]. Conventional models need the global and complete information of the data center, and derive cost based on the usage of resources [14,15]. The models are mostly centralized in nature. Economic and game-theoretic models for resource allocation derive cost based on the value that the customer derives from their demands and the available services. Such models are not only decentralized but also offer incentives to customers, and hence have recently aroused extensive attention in industrial world and academe.

Among the studies employing game theory to investigate the resource allocation problems in clouds, several works [18,10,11] focused on the system performance or the tradeoff between the efficiency of resources allocation and energy consumption, and adopted the decentralized allocation strategy to ease the burden of system management. Although these schemes exhibited significant improvement in the management of resource allocation, they still gave insufficient focus on the economic incentive of both the customers and the providers. For example, the customer's valuation for services was not taken into account, and CPs can accept all user requests regardless of the operating cost and the customer's valuation.

Auction-based approaches have been explored in a series of work in recent cloud computing literatures, due they require little global information, are decentralized, and easy to be implemented in distributed systems. Di Valerio et al. [19] formulated the service provisioning problem as a *Stackelberg* game, and computed the equilibrium price and allocation strategy by solving the associated optimization problem. Zhang et al. [20] studied the resource allocation problem with real-time demand arrivals, and propose a truthful online auction-based allocation policy. Zaman et al. [12] proposed an auction-based VM allocation mechanism, which is approximately efficient and generates higher revenue than the currently used fixed-price mechanisms for allocating VMs. In their extended work [13], the set of VMs is selected in a dynamic fashion that reflects the market fluctuation over time. In the above studies, the provisioning VMs are regarded as type-oblivious commodities to respond to the diversity of VMs. The authors of [21,22] employed the concept of bid density to measure the quantity of the VMs bundle requested by the customer. However, they did not normalize the numerical representation for different types of resources, such as vCPU, memory and storage, to ensure the fairness of the evaluation. Wang et al. [23] proposed a fitness-enabled auction,

in which the key is to require a standard machine as evaluation benchmark to address the VM heterogeneity in cloud computing. However, finding a standard machine in practice is difficult.

Truthfulness, also known as strategy-proof or incentive-compatible, is an essential property that avoids market manipulation and ensures auction fairness and efficiency in economic market. As the only type of auctions that can simultaneously guarantee truthfulness and economic efficiency, the VCG mechanism has obtained widely attention for the resource allocation in cloud markets. However, VCG-based mechanisms suffer from computational complexity because the winner determination is an NP-hard problem [2]. Thus, to achieve truthfulness and reduce computing complexity, numerous schemes [24,20,12,13,21,22] were proposed by extending classic VCG mechanism, in which they first designed an approximation algorithm or dynamic programming-based algorithm and then resorted to the critical bid rule to calculate sale price for each user. The authors of [24–26] utilized the primal–dual optimization algorithms and randomized reduction techniques to design a set of truthful, polynomial-time auctions based on VCG for dynamic resource provisioning, which is computationally efficient and truthful in expectation. Tanaka et al. [27] proposed a dynamic programming-based algorithm for service selection and VCG payment calculation. Mihailescu et al. [28] introduced a reverse auction model for the selection of services in cloud platforms to exploit dynamic pricing by assuring truthfulness. The studies aimed to maximize the utility of users and providers and reduce the computational cost while guaranteeing the truthfulness of the mechanisms proposed by them. These schemes shown significant improvement but disregarded the envy-free property and the over-supplied scenario, which may result in the instability of the system and the decrease of the provider's revenue.

Envy-freeness induces the notion of a fair allocation which can ensure that the auction mechanism is stable, such that no bidder will be happier with the outcome of another bidder after the auction is run. Hence, numerous studies [17,29–33] in various fields viewed it as an in-ignorable attribute while implementing the mechanism of resources allocation. The analysis of [29–33] relied on the assumption that the available items, such as digital goods and VMs, sold to customers were homogeneous commodities, which are not suitable to the cloud market that provides multiple types of resources to users. Toosi et al. [17] extended the above investigations and proposed an auction-based mechanism called Online Ex-CORE that can generate near optimal profit for the cloud provider in a single round of auction. They constructed an auction-based mechanism that is envy-free and truthful with high probability on price dimension. Although similar technologies are used in our work, the difference from their work is that we consider that the scenario of VMs provisioning is various types rather than single type. Furthermore, the mechanism proposed by us is envy-free and fully truthful on both of price and quantity dimensions.

3. System model

In this section, we introduce the system model and the problems formulation. For the convenience of the readers, the major notations used in this paper are listed in Table 1.

3.1. Normalizing VMs

In cloud resource markets, commodities exchanged are various types of VMs which are built by distinct types of resources, such as the virtual CPU (vCPU) with the number of cores, the storage and memory with G Byte. Hence, We may encounter the inconsistency of measurement while conducting comparison and counting the amount of the resources consumed by different types of VMs.

Table 1
Notations.

Symbol	Meaning
i	Subscript of customers
j	Subscript of VMs
k	Subscript of resource types
VM_j	VM of type j
r_j^k	Amount of type k resource consumed by VM_j
N_j^k	Scaled value of type k resource of VM_j
R_{max}^k	Maximum amount of type k resource assigned to a VM in all VMs
σ_k	Weighting value of R_{max}^k amount of type k resource
N_j	Normalized value of total resources consumed by VM_j
C_k	Available capacity of type k resources in the system
O_i	Order of the customer i
O	Set of all customers' orders
b_i	Bid of the customer i for her order
b_{-i}	Bid set of all customers without the customer i
Q_i	Normalized number of VMs ordered by the customer i
q_i^j	Number of VM_j requested by the customer i
D_i	Set of the VMs ordered by the customer i
$V_i(D_i)$	customer i's true valuation for resource set D_i
p	Set of the unit price for which the customers must pay
p_i	Unit price for which the customer i must pay
U_i	Customer i's utility
π	Permitted supremum of VM units ordered by a customer
ξ	Maximum VM units requested by a user in the submitted orders
S	Aggregated units of sold VMs
F	Target revenue
$\hat{p}_i$	Payment of user i

Accordingly, only considering the cores of the CPU regardless of the fact that cloud providers also offer matching resources for memory and storage is unreasonable and unfair. Inspired by the multiple criteria decision making (MCDM) technique [34], we first scale and normalize the amount of each type of resource of VMs before performing the resource allocation in clouds.

Assume a cloud provider offers m types of VMs, which are denoted by set $M = \{VM_1, VM_2, \ldots, VM_m\}$. Each VM comprises k different types of resources, $k \in \{1, 2, \ldots, K\}$. Let r_j^k denote the amount of type k resource assigned to VM_j. The scaled value N_j^k for type k resource of VM_j is given by

$$N_j^k = \frac{r_j^k}{R_{max}^k}, \tag{1}$$

where $R_{max}^k \triangleq \max_{j \in M} r_j^k$ denotes the maximum amount of the type k resource observed across all the VMs provisioned by the cloud provider. Therefore, we have $0 < N_j^k \leq 1$ for $\forall j \in j, \ldots, m$.

Additionally, to capture the cost level of distinct type resources, it is necessary to assign a suitable weighting value for every type resource. The Analytic Hierarchy Process (AHP) [35] is a well-known technique used in these kinds of complex situations. In our work, the AHP is used to measure the cost level for distinct type resources based on the provider's criteria, such as operating costs, investment costs and so on. We construct a matrix $\Gamma = \{a_{kl};\ 1 \leq k \leq K, 1 \leq l \leq K\}$ using the AHP, where a_{kl} denotes the pairwise comparison that indicates how many times more important R_{max}^k amount of type k resource is over that of type l resource. Then, the weighting value σ_k for R_{max}^k amount of type k resource is determined by

$$\sigma_k = \frac{(\prod_{l=1}^{K} a_{kl})^{\frac{1}{K}}}{\sum_{k=1}^{K}(\prod_{l=1}^{K} a_{kl})^{\frac{1}{K}}}. \tag{2}$$

Obviously, there are $0 < \sigma_k \leq 1$ and $\sum_{k \in K} \sigma_k = 1$.

Next, using simple additive weighting, the normalize value N_j for the aggregated resources assigned to VM_j is calculated by

$$N_j = \sum_{k=1}^{K} N_j^k \sigma_k, \ \forall j \in j, \ldots, m. \tag{3}$$

Table 2
VM instance types offered by Amazon EC2.

Type	Small	Medium	Large	Extra-large
vCPU(core)	1	1	4	8
Memory(GB)	1.7	3.75	7.5	15
Storage(GB)	160	410	850	1690

For example, the four types of VMs offered by Amazon EC2 are presented in Table 2, and vCPU represents Type 1 resource, memory, Type 2 resource, storage, Type 3 resource. The maximum amount of vCPU, memory, and storage is 8, 15, and 1690 respectively (Table 2). Let 3-tuple $\langle \sigma_1, \sigma_2, \sigma_3 \rangle$ represent their weighting values. The normalized values of *Small* ($j = 1$) and *Medium* ($j = 2$) VM can be computed by $N_1 = 1\sigma_1/8 + 1.7\sigma_2/15 + 160\sigma_3/1690$ and $N_2 = 2\sigma_1/8 + 3.75\sigma_2/15 + 410\sigma_3/1690$, respectively.

In good agreement with the cloud's promise of delivering an unlimited supply of resources, the provider's capacity far exceeds the total demand of the users in most cases. Nevertheless, the capacity of a cloud data center still is constrained in fact. We assume that C_k denotes the available capacity of type k resources at a given time slot in the data center.

3.2. Customer utility formulation

Suppose n customers requesting computing resources from the cloud provider at time t. Each customer can submit an order (request) as a vector $O_i = (D_i, b_i)$, which specifies the number of VMs and her bid, $D_i = \langle q_i^1, q_i^2, \ldots, q_i^m \rangle$ denotes the bundle of VMs requested by the customer i for different types of VMs, q_i^j represents the number of type VM_j, and b_i represents the maximum price that the customer i is willing to pay for using the requested bundle for a unit time. For example, $O_i = (\langle 2, 8, 5, 1 \rangle, \$26)$ represents the order of the customer i, who requests two *Small* VMs, eight *Medium* VMs, five *Large* VMs and one *Extra-Large* VM, and her bid is \$26.

According to Eq. (3), the quantity of the VMs ordered by the customer i can be converted into the following normalized value.

$$Q_i = \sum_{j=1}^{m} N_j q_i^j. \tag{4}$$

Throughout the paper, we will utilize Eq. (4) to count the total number of the VMs requested by the customer i. In such ways, the comparison of the VM bundles ordered by different customers can be fairly conducted under a common decision making technique.

In our model, the customers are *single-minded*. That is, the partial fulfillment of requests, in which only a fraction of the number of VMs requested is allocated to the customer, is not accepted by the customers. Moreover, the customers are not aware of the future and have no time-dependent valuation for resources, $p = (p_1, \ldots, p_i, \ldots, p_n)$ denotes the set of the unit price for which the customers must pay, $b = (b_1, \ldots, b_i, \ldots, b_n)$ is the set of all users' bids, and $V_i(D_i)$ represents the customer i's true valuation for the resource set D_i. We define the customer i's utility at time t for one time slot of VM usage as follows:

$$U_i(D_i, b) = (V_i(D_i) - p_i Q_i) x_i, \tag{5}$$

subject to

$$0 \le p_i Q_i \le b_i \le V_i(D_i), \ \forall i \in \{1, 2, \ldots, n\},$$

where the allocation vector $x = (x_1, x_2, \ldots, x_n)$ represents the decision variable, its *ith* component x_i is defined as follows:

$$x_i = \begin{cases} 1 & \text{if } D_i \text{ is assigned to customer } i, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

A customer for which $x_i = 1$ is called a *winner* and pays the corresponding price $p_i Q_i$, and then acquires the requested bundle D_i; otherwise, the customer is called a *loser* and does not make any payment to the cloud provider, and no resource is assigned to her.

Furthermore, the customers are assumed to be individually rational, which means that no client has negative expected utility for taking part in the mechanism. Hence, as long as a customer is deemed beneficial, she will strategically misreport her bid b_i and the bundle of demanded VMs D_i' (i.e., $b_i \ne V_i(D_i)$ or $D_i' \ne D_i$) in order to maximize her utility, where $V_i(D_i)$ and D_i are the private information known only by herself.

Truthfulness is capable to avoid market manipulation and ensure auction fairness and efficiency. Besides, truthfulness can simplify the strategic decision process for all clients, and the client can report her true valuation irrespective of other client's valuation [2]. Hence, the truthfulness is taken into account in our models and described as follows.

Definition 3.1 (*Truthfulness*)**.** A mechanism is truthful, if any customer reports her truthful valuation b_i^*, for any $b_i \ne b_i^*$ and any valuation profile of others b_{-i}, her utility $U_i(D_i, (b_i^*, b_{-i})) \ge U_i(D_i, (b_i, b_{-i}))$ always holds.

3.3. Formulation of VM allocation and pricing

In fact, the optimal bidding strategy of each customer is to maximize her utility function in Eq. (5). Similarly, the objective of the cloud provider is to maximize her profit generated by the function $A(p, x)$ formulated as follows:

$$A(p, x) = \sum_{i=1}^{n} Q_i p_i x_i - c(x), \tag{7}$$

where $c(x)$ is an inherent cost in producing the outcome x which must be paid by the cloud provider, it is typically viewed as a constant.

Therefore, we formulate the problem of VMs pricing and allocation in clouds as an integer program as follows:

$$\text{maximize } A(p, x), \tag{8}$$

subject to

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_i q_i^j r_j^k < C_k, \ \forall k \in K, \tag{9}$$

$$x_i \in \{0, 1\}, \ \forall i \in \{1, 2, \ldots, n\}, \tag{10}$$

$$0 \le p_i Q_i \le b_i x_i, \ \forall i \in \{1, 2, \ldots, n\}. \tag{11}$$

The solution of the above problem consists of an allocation vector $x = (x_1, x_2, \ldots, x_n)$, that maximizes the cloud providers revenue and the unit price $p = (p_1, p_2, \ldots, p_n)$ for each customer i that requests her bundle D_i. Constraints (9) ensure that the allocation of each resource type does not exceed the available capacity of the cloud provider's resources. Constraints (10) represent the integrality requirements for the decision variables, which satisfy the case wherein the customers are *single-minded*. Constraints (11) promise that the customer's payment charged by the cloud provider is not greater than her bid. Notably, our models assume that the information about all customers orders is available, and the information about the future orders of the customers is unavailable at time of solving it.

4. Mechanism design framework

In this section, we employ the combinatorial auction scheme to design an mechanism promising the properties of truthfulness and envy-freeness. Fig. 1 presents the mechanism framework proposed by us, which will be elaborated in Section 4.3.

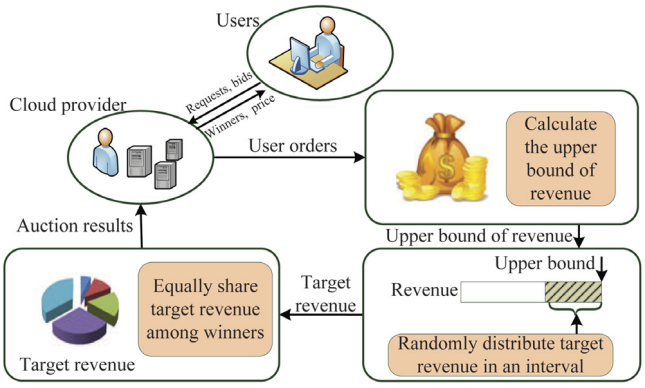


Fig. 1. Mechanism framework.

Algorithm 1 EFM(O, $C_{available}$): Framework of envy-free mechanism

Input:
the orders of the users: $O = \{O_1, ..., O_n\}$,
the vector of available resource capacities: $C_{available} = (C^1_{available}, \ldots, C^K_{available})$.

Output:
the payment vector for the users: $\hat{P} = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n)$,
the allocation vector for the users: $x = (x_1, x_2, \ldots, x_n)$.

1: Sort the customers in descending order of their bid densities, $b_1/Q_1 \geq b_2/Q_2 \ldots \geq b_n/Q_n$
2: Determine the upper bound the revenue $G(O)$
3: Calculate the target revenue F with a consensus estimate $f(.)$ on $G(O)$
4: Determine the payment vector $\hat{P}$ and the allocation vector x with *RevenueExtractor*(F, O)

4.1. Envy-free property

The mechanisms, in which the customers are charged at distinct prices even though identical services or resources are provisioned to them, are unstable and unfair. The losing customer envies the winning customer's outcome. Hence, designing a practical auction mechanism should consider not only the truthful property but also the envy-free property, which has been suggested as a highly desirable property of auction mechanism in the literatures [2,17,30,32]. The definition of envy-freeness is described as follows.

Definition 4.1 (*Envy-Freeness*)**.** An auction-based mechanism is said to be envy-free if for every bidders i, i', it holds that:

$$V_i(D_i) - p_i Q_i \geq V_i(D_{i'}) - p_{i'} Q_{i'}$$

Envy-freeness is capable to provide equal treatment among bidders and induces the perception of a fair allocation. No bidder in an envy-free auction can increase her utility by adopting another bidder's outcome. It is clear that the auction using a single sale price p is envy-free, in which all bidders with bid value greater than p win, whereas all bidders with bid value lower than p lose. In our schemes, we attempt to follow such a strategy to implement the envy-freeness of mechanisms.

4.2. Upper bound of revenue

To achieve the envy-freeness and the revenue maximization of CPs simultaneously, we introduce the optimal single-priced auction to calculate the upper bound of CPs' revenue, in which goods are sold to the winners at an identical unit price. It is defined as follows.

Definition 4.2. Given that the customers are sorted in descending order by their bid densities, the upper bound of revenue with at least two winners is decided by

$$G(O) = \max_{L} \frac{b_L}{Q_L} \sum_{i=1}^{L} Q_i, \tag{12}$$

where $O = \{O_1, \ldots, O_n\}$ represents the order set of the users, and b_L/Q_L is the *Lth* largest user's bid density, that is, how much she is willing to pay for using one normalized VM in a time slot. Obviously, $G(O)$ is the maximum revenue achieved by the service provider at the user-affordable price b_L/Q_L, and thus viewed as the upper bound of the available revenue.

The mechanism constructed by above definition has a single sale price b_L/Q_L and thus is envy-free. Each customer i with $b_i/Q_i \geq b_L/Q_L$ wins at unit price b_L/Q_L, and then is charged $(b_L/Q_L)Q_i$. Notably, this mechanism is untruthful because customers can misreport their bid to obtain a more favorable outcome for them. Therefore, the question arises as to how a single price can be computed for an order-independent auction while approaching the upper bound of the revenue.

4.3. Framework of envy-free mechanism

The *EFM* mechanism is proposed and briefly summarized in Algorithm 1 and Fig. 1. It combines two general ideas. The first is *consensus estimate* that picks a randomized function $f(.)$ that estimates the upper bound of the revenue and achieves consensus with high quality [2,32]. The objective of using this idea is to avoid market manipulation.

Let $O_{-i} = \{O_1, \ldots, O_{i-1}, O_{i+1}, \ldots, O_n\}$ denotes the order set of all users without the user i. Assume there is some $\rho > 1$ such that for each customer i, $G(O_{-i}) \in [G(O)/\rho, G(O)]$, the *consensus estimate* is described as follows.

Definition 4.3 (*Consensus Estimate*)**.** A randomized function $f(.)$ is a consensus estimate of $G(O) > 0$ if for any $G(O_{-i}) \in [G(O)/\rho, G(O)]$ satisfies $f(G(O)) = f(G(O_{-i}))$.

The second is *RevenueExtraction* that is a special case of the Moulin–Shenker cost sharing mechanism [36], which has proven to be truthful and envy-free. Given the customers sorted in descending order by the bid density of each user b_i/Q_i and the target revenue F, the *RevenueExtraction* finds the largest L such that the *Lth* highest customer's bid density $b_L/Q_L \geq F/\sum_{i=1}^{L} Q_i$, then the highest L customers equally share the target revenue F. In our schemes, the quantity of goods requested by the customer is extended to multiple units rather than one unit in the original algorithm.

In the *EFM* mechanism, first all customers are sorted by the bid density of each user b_i/Q_i, and then $G(O)$ (the upper bound of the revenue) is drawn. Next, it estimates $G(O)$ with a randomized function $f(.)$, and acquires the target revenue F. Finally, the *RevenueExtractor* finds the highest L customers (*winners*), $x_i = 1$, $i \leq L$, and charges $\hat{p}_i = Q_i(F/\sum_{i=1}^{L} Q_i)$ for user i. Otherwise 0. The mechanism is performed periodically by the cloud provider.

5. Envy-free mechanism for VM allocation and pricing

In this section, we present several key technologies implementing *EFM*, such as, determining the randomized function $f(.)$ and its related parameters. Moreover, the detailed procedures of Steps 2 to 4 in *EFM* are given with several algorithms.

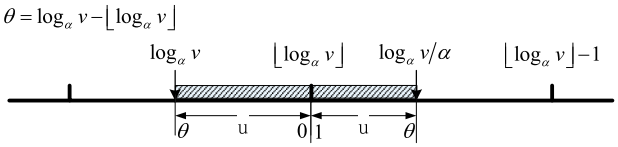


Fig. 2. Probability distribution of $\lfloor (\log_\alpha v - u) \rfloor + u$.

5.1. Consensus estimate

Here, we are interested in a function that provides a sufficiently accurate estimate of $G(O)$ that is constant on $G_{-i}(O)$ for all i (i.e., it achieves consensus). If $G_{-i}(O)$ could be limited by a constant fraction of $G(O)$, it is possible to capture a good estimate of $G(O)$. Accordingly, the parameter ρ for achieving such a function needs to be decided.

In our model, to prevent customer collusion, in which the customers collaborate to gain an unfair market advantage or limit fair competition, the supremum of the aggregated number of the VMs requested by arbitrary customer is set to a constant π. Let $\xi \leq \pi$ denotes the maximum quantity of VMs requested by the customer and is given as follows according to Eq. (4):

$$\xi = \max(Q_1, Q_2, \ldots, Q_n). \tag{13}$$

Removing any one order can change $G(O)$ to at most a factor of $(S - \xi)/S$, when the quantity of sold items in the auction is S. For the case that $G_{-i}(O)$ is a constant fraction of $G(O)$, let $\rho = S/(S-\xi)$, the following conclusion is hold

$$\frac{1}{\rho} G(O) \leq G(O_{-i}) \leq G(O) \tag{14}$$

Next, we need to seek a function $f(.)$ obeying a distribution of functions that estimates $G(O)$ with high quality and achieves consensus with high probability. Inspired by reference [32], the randomized function $f(.)$ for any $v \in [G(O)/\alpha, G(O)]$ is formally formulated as follows:

$$f(v) = \alpha^{\lfloor \log_\alpha v - u \rfloor + u}, \tag{15}$$

where u is a constant chosen uniformly on [0,1], and $\alpha > \rho$ is a constant picked to maximize the quality of the estimate. Fig. 2 describes the probability distribution of $\lfloor (\log_\alpha v - u) \rfloor + u$ as varying u on [0,1]. It is clear that the randomized function $f(v)$ obeys the distribution of functions of the form $\alpha^{u-1} v$ with u chosen uniformly on [0,1]. Therefore, we have following conclusion:

Lemma 5.1. *Randomized function $f(G(O))$ is a consensus estimate for any $G(O_{-i}) \in [G(O)/\rho, G(O)]$ if $f(G(O)) \leq G(O)/\rho$.*

Proof. Consider that $f(G(O))$ is distributed identically to $\alpha^{u-1} G(O)$ for u uniform on [0,1]. Hence, for any $G(O)$, there is

$$G(O)/\alpha \leq f(G(O)) \leq G(O).$$

Additionally, $G(O)/\rho$ is the lower bound of $G(O_{-i})$ according to Eq. (14), we therefore have

$$G(O)/(\rho\alpha) \leq f(G(O)/\rho) \leq G(O)/\rho.$$

Fig. 3 gives more details for the probability distribution of exponent fraction of the function when $v = G(O)$ and $v = G(O)/\rho$. The second sub-figure and the third sub-figure characterize the probability distributions for $log_\alpha(G(O)/\rho) \geq \lfloor log_\alpha G(O) \rfloor$ and $log_\alpha(G(O)/\rho) < \lfloor log_\alpha G(O) \rfloor$ respectively. Obviously, to ensure the existing of consensus estimate for any $G(O_{-i}) \in [G(O)/\rho, G(O)]$, the following inequality must be met

$$\lfloor log_\alpha(G(O) - u) \rfloor + u \leq \log_\alpha(G(O)/\rho),$$

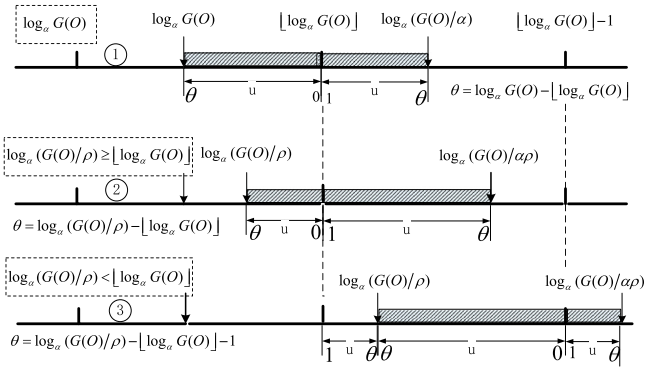


Fig. 3. Probability distribution of $\lfloor (\log_\alpha v - u) \rfloor + u$ for ①$v = G(O)$ and ②③$v = G(O)/\rho$.

such that there is $u \in [0, 1]$ for any user i and $G(O - i)$, which allow $\lfloor log_\alpha G(O) - u \rfloor + u = \lfloor \log_\alpha G(O_{-i}) - u \rfloor + u$ to be satisfied. Using above conclusion, we hence can say the randomized function $f(G(O))$ is a consensus estimate, if

$$f(G(O)) \leq G(O)/\rho.$$

This remark completes the proof.

Given $f(G(O))$ is a randomized function and using its estimate value as the target revenue, it is reasonable to consider the expected value of $f(G(O))$ for achieving high payoff. Furthermore, the probability density function for $\alpha^u \leq x$ is $1/(x \ln \alpha)$ for $1 \leq x \leq \alpha$. Hence, given $f(G(O)) \in [G(O)/\alpha, G(O)/\rho]$ and referred to [32], the expected value is described as follows:

$$E(f(G(O))) = \frac{G(O)}{\ln \alpha} \left(\frac{1}{\rho} - \frac{1}{\alpha} \right). \tag{16}$$

5.2. Picking parameter

Following the aforementioned discussions, the choice of parameter α is considerably significant in the implementation of our proposed mechanism. To consensus to work on the order set O on which at least S items are sold, we need $\alpha > \rho = S/(S - \xi)$. Otherwise, no consensus is achievable. Considering a large cloud market, as $S \to \infty$, we have

$$\begin{aligned} \lim_{S\to\infty} E(f(G(O))) &= \lim_{S\to\infty} \frac{G(O)}{\ln \alpha} \left(\frac{1}{\rho} - \frac{1}{\alpha} \right) \\ &= \lim_{S\to\infty} \frac{G(O)}{\ln \alpha} \left(\frac{S - \xi}{S} - \frac{1}{\alpha} \right) \\ &= \lim_{S\to\infty} \frac{G(O)}{\ln \alpha} \left(1 - \frac{1}{\alpha} \right) \\ &= \lim_{S\to\infty} \frac{G(O)}{\ln \alpha \left(1 + \frac{1}{\alpha - 1} \right)}. \end{aligned} \tag{17}$$

Let $\alpha = 1 + t$, $t = \frac{\tau\xi}{S-\xi}$, and τ is a constant, $\tau > 1$, $\tau \in \Re$, then Eq. (17) can be converted as follows:

$$\lim_{S\to\infty} E(f(G(O))) = \lim_{S\to\infty} \frac{G(O)}{\left(\ln(1 + t) + \ln(1 + t)^{\frac{1}{t}} \right)}. \tag{18}$$

For $S \to \infty$ and fixed $\xi \ll S$, we have $t \to 0$. This means that

$$\begin{aligned} \lim_{S\to\infty} E(f(G(O))) &= \lim_{S\to\infty} \frac{G(O)}{(\ln 1 + \ln e)} \\ &= G(O). \end{aligned} \tag{19}$$

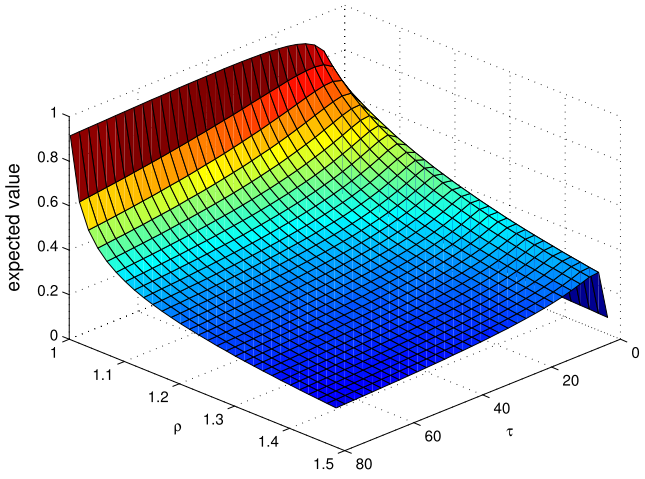


Fig. 4. Expectation, τ and ρ.

It is clear that $\alpha = 1+\frac{\tau\xi}{S-\xi} > \rho = 1+\frac{\xi}{S-\xi}$, if $\tau > 1,\ \tau \in \Re$, and ensures achieving a consensus for the mechanism. Furthermore, just doing so can make the expectation approach the upper bound of revenue as $S \to \infty$. Hence, it is appropriate that parameter α is set to such value for our mechanism.

In the following, we will discuss how to determine the value of τ. For $\rho = 1 + \frac{\xi}{S-\xi}$, we have $t = \tau(\rho - 1)$. Accordingly, α can be rewritten as follows:

$$\alpha = 1 + \tau(\rho - 1). \tag{20}$$

Based on above equation, Eq. (16) can be rewritten as follows:

$$E(f(G(O))) = \frac{G(O)}{\ln(1+\tau(\rho-1))}\left(\frac{1}{\rho} - \frac{1}{1+\tau(\rho-1)}\right). \tag{21}$$

Fig. 4 shows the affection of varying τ and ρ on the ratio of the expected value to $G(O)$. It is obvious that the ratio is concave if ρ is fixed. Hence, given ρ, τ maximizing the expected payoff can be determined by numerical methods such as binary searching. Furthermore, the ratio is close to 1 as ρ declines. That is, the expected payoff can approach the optimal value when the cloud market is enough large.

5.3. Determining the upper bound of revenue

The procedure of determining the upper bound of revenue is elaborated in Algorithm 2 (*DUBRE*) which correspond to Steps 1 and 2 of *EFM*. Cloud providers typically preset a reserve price for their resources to ensure that their incomes are not less than the operating cost. Let C_R and C_I be the costs associated with running, respectively, idling a unit VM instance for one unit of time. A distinct observation is that $C_R - C_I$ is the critical price running a unit VM by the cloud providers. Hence, the reserve price is set to $C_{res} = C_R - C_I$ in *DUBRE* algorithm (line 2).

DUBRE calculates the bidding density for each user and then removes the users for which $b_i/Q_i < C_{res}$, such that the accepted users pay at least the reserve price. The users with the quantity of request VMs greater than the permitted value also are discarded (lines 3–8). Then, *DUBRE* algorithm under the available resource capacities decides the set of acceptable users' orders A in decreasing order of the users' bidding densities b_i/Q_i(lines 10–24). Notably, $C^k_{available}$ denotes the available capacities for the k type of resource, such as CPU, memory and storage. *DUBRE* measures whether it can implement the request of the user i (lines 11–19). If there are insufficient resources, user i will be rejected, otherwise,

Algorithm 2 DUBRE(O, $C_{available}$): Determining the upper bound of revenue

Input:
the orders of all customers: $O = \{O_1, ..., O_n\}$,
the vector of available resources capacities: $C_{available} = (C^1_{available}, \ldots, C^K_{available})$.
Output:
the upper bound of revenue: P,
the set of acceptable user's order: A,
the sum of the VMs determined by *DUBRE*: S.

1: $A \leftarrow \phi$
2: $C_{res} \leftarrow C_R - C_I$ //set the reserve price
3: **for** *all* $O_i \in O$ **do**
4: $Q_i \leftarrow \sum_j^m N_j q_i^j$
5: $d_i \leftarrow b_i/Q_i$
6: **if** $d_i < C_{res}$ *or* $Q_i > \pi$ **then**
7: $O \leftarrow O \backslash \{O_i\}$
8: **end if**
9: **end for**
10: Sort the set O in non-increasing order of d_i
11: **for** *all* $O_i \in O$ **do**
12: $\hat{C}_{available} \leftarrow C_{available}$
13: $flag \leftarrow true$
14: **for** *all* $k \in K$ **do**
15: $\hat{C}^k_{available} \leftarrow \hat{C}^k_{available} - \sum_{j=1}^m q_i^j r_j^k$
16: **if** $\hat{C}^k_{available} < 0$ **then**
17: $flag \leftarrow false$
18: **break**
19: **end if**
20: **end for**
21: **if** *flag* **then**
22: $C_{available} \leftarrow \hat{C}_{available}$
23: $A \leftarrow A \cup \{O_i\}$
24: **end if**
25: **end for**
26: $P \leftarrow \max_L \ \frac{b_L}{Q_L}\sum_{i=1}^{L} Q_i,\ \forall O_i \in A$
27: $S \leftarrow \arg\max_{\sum_{i=1}^{L} Q_i} \ \frac{b_L}{Q_L}\sum_{i=1}^{L} Q_i,\quad \forall O_i \in A$
28: **return** $P,\ A,\ S$

she will be allocated and the amount of available resources will be updated (lines 23 and 24). Finally, according to Definition 4.2, *DUBRE* draws the upper bound of revenue P and corresponding overall number of the sold VMs S (lines 26 and 27). P, S and the set of acceptable users' orders A are returned to the mechanism.

5.4. Calculating target revenue

Having obtained the upper bound of revenue for the set of orders submitted by users, we proceed to calculate the target revenue corresponding to Step 3 of *EFM*. The related details are elaborated in Algorithm 3 (*CATAR*). The essence of *CATAR* is to fulfill the randomized function $f(.)$ formulated in Eq. (15), such that the designed mechanism satisfies the properties of both envy-freeness and truthfulness. Its input parameters, such as P, the upper bound of revenue, A, the set of acceptable user order, and S, the sum of requested instances in the set A, can be acquired by executing *DUBRE* algorithm.

In *CATAR* algorithm, the parameters for the randomized function $f(.)$ is first determined (lines 1–4). Next, u is chosen uniformly on [0,1] and then the target revenue F is calculated by Eq. (15).

Algorithm 3 CATAR(P, A, S): Calculating the target revenue

Input:
the upper bound of revenue: P,
the set of acceptable user's order: A,
the sum of the VMs selected by *DUBRE*: S.
Output:
the target revenue: F.
1: $\xi \leftarrow \max(Q_1, Q_2, ..., Q_{|A|})$
2: $\rho \leftarrow \frac{S}{S-\xi}$
3: find τ maximizing Eq. (21)
4: $\alpha \leftarrow 1 + \frac{\tau\xi}{S-\xi}$
5: **repeat**
6: $u \leftarrow random(0, 1)$
7: $F \leftarrow \alpha^{\lfloor \log_\alpha P-u \rfloor + u}$
8: **until** $F \leq P/\rho$
9: **return** F

The algorithm ends when the target revenue F is not more than P/ρ (lines 5–8).

5.5. Determining payment and allocation

The approach of resource allocation and payment for users is presented in Algorithm 4 (*RevenueExtractor*), which is based on the cost-sharing mechanism [36]. In this algorithm, given the target revenue F and the sorted orders A, the *RevenueExtractor* finds the largest L such that the sale price is not greater than the *Lth* user's bidding density (lines 3–10). The target revenue F is then shared among the L users (winners) based on the number of VMs requested by them, that is, each of these users are charged the same sale price $\delta = F/\sum_{i=1}^{L}\sum_{j=1}^{m} N_j q_i^j$ for using one unit normalized VM in a time slot (Line 11). If the user i is one of the *winners*, her total payment is computed by $\delta\sum_{j=1}^{m} N_j q_i^j$, and her allocation variable x_i is set to 1 (lines 12–15). Otherwise, her payment and allocation variable are equal to 0 in the auction. In doing so, the final sale price is order-independent and single such that the properties of truthfulness and envy-freeness is guaranteed.

5.6. Properties of EFM

In this subsection, we investigate the properties of *EFM*, and show that the mechanism is truthful and envy-free. We first show that it is individual rational.

Theorem 5.1. *EFM is individual rational.*

Proof. Given user i as a winning user, we need to prove that if user i reports her true request then her utility is non-negative. It is easily seen that the sale price for using one unit VM is not greater than the winning user's bidding density in lines 5–9 of Algorithm 4. Hence, the utility of user i (e.g., $U_i = b_i - \hat{p}_i \geq 0$) is non-negative. Additionally, a truthful user who does not win is not incurring a loss due to she obtains 0 utility. This proves the individual-rationality of *EFM*.

RevenueExtractor is a significant part of *EFM*, We now prove that it is truthful and envy-free for *single-minded* users.

Lemma 5.2. *RevenueExtractor is truthful and envy-free for* single-minded *users.*

Proof. We first prove the property of envy-freeness. It is obvious that the sale price produced by *RevenueExtractor* for using one unit VM is single. In terms of Definition 4.1, no bidder in the mechanism

Algorithm 4 RevenueExtractor(F, A): Determining the VMs allocation and payment

Input:
the set of acceptable user's order: A,
the target revenue: F.
Output:
the payment vector for each users: $\hat{P} = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n)$,
the allocation vector for each users: $x = (x_1, x_2, \ldots, x_n)$.
1: $\hat{P} \leftarrow (0, \cdots, 0)$
2: $x \leftarrow (0, \cdots, 0)$
3: $sum \leftarrow \sum_{i=1}^{|A|}\sum_{j=1}^{m} N_j q_i^j$
4: **for** $L \leftarrow |A|$ to 1 **do**
5: **if** $\frac{F}{sum} > \frac{b_L}{\sum_{j=1}^{m} N_j q_L^j}$ **then**
6: $sum \leftarrow sum - \sum_{j=1}^{m} N_j q_L^j$
7: **else**
8: **break**
9: **end if**
10: **end for**
11: $\delta \leftarrow \frac{F}{sum}$
12: **for** $i \leftarrow 1$ to L **do**
13: $\hat{p}_i \leftarrow \delta\sum_{j=1}^{m} N_j q_i^j$
14: $x_i \leftarrow 1$
15: **end for**
16: **return** $\hat{P}, x$

can increase her utility by adopting another bidder's outcome. Hence, *RevenueExtractor* is envy-free.

Next, we will prove that no bidder can increase her utility by reporting any false bid value or request quantity. Here, there are two situations that need to be considered respectively. The first is that the user varies only her bid value to increase the utility. Let L be the number of winning users. *RevenueExtractor* in this case shares the target revenue F among the winners, and charges each winner at the same sale price $\delta = F/\sum_{i=1}^{L}\sum_{j=1}^{m} N_j q_i^j$ for using one unit normalized VM in a time slot. Hence, it can be regarded as a special case of the Moulin–Shenker cost sharing mechanism [36] that has been proven to be truthful. Accordingly, we can say *RevenueExtractor* is truthful for the price dimension.

The second scenario is that the user varies only her request quantity to increase the utility. Specially, in our model, the users are *single-minded*. That is, the partial fulfillment of requests is not accepted by the user. Therefore, the request quantity reported by the user is not less than her true request quantity. Firstly, assume user i is a loser, reporting more request quantity means lower bid density, thus she is still a loser and her utility is 0. Just doing so cannot increase her utility.

Whereas, let user i is a winner, $i \leq L$, Q_i^* is her true request quantity and Q_{-i} denotes the total request quantity of the L winners without user i. We have following conclusions:

$$\frac{F}{\sum_{i=1}^{L}\sum_{j=1}^{m} N_j q_i^j} = \frac{F}{Q_{-i} + Q_i^*} \leq \frac{b_L}{\sum_{j=1}^{m} N_j q_L^j} \tag{22}$$

and

$$\frac{F}{Q_{-i} + Q_i^* + Q_{L+1}} > \frac{b_{L+1}}{\sum_{j=1}^{m} N_j q_{L+1}^j} \tag{23}$$

Eq. (22) implies that the L winners' bid densities are not less than the sale price for using one unit VM. Assume user i is still a winner when she reports her request quantity $Q_i^* + x$, $x \geq 0$. If $x \leq Q_{L+1}$, we know that the member of the winners will not change by observing Eqs. (22) and (23) . Hence, the payment of user i is

given as follows:

$$\hat{p}_i = \frac{F(Q_i^* + x)}{Q_{-i} + Q_i^* + x}. \tag{24}$$

The derivative of the above equation on x is:

$$\frac{d\hat{p}_i}{dx} = \frac{FQ_{-i}}{(Q_{-i} + Q_i^* + x)^2}. \tag{25}$$

If $x > Q_{L+1}$ and she is still a winner, then more users may become winners. It is obvious that x is not less than the aggregated request quantity of those new winners. Accordingly, the payment of user i is near to following expression:

$$\hat{p}_i = \frac{F(Q_i^* + x)}{Q_{-i} + Q_i^* + 2x}. \tag{26}$$

The derivative of the above equation on x is:

$$\frac{d\hat{p}_i}{dx} = \frac{F(Q_{-i} - Q_i^*)}{(Q_{-i} + Q_i^* + 2x)^2}. \tag{27}$$

Obviously, Eq. (25) is positive. Moreover, as the cloud market is enough large, we have $Q_{-i} \gg Q_i^*$. Thus Eq. (27) also is positive. They mean that user i has to pay more and then her utility is reduced when the request quantity reported by her is greater than her true demand.

Additionally, due to the increase in the request quantity leads to a decline in the bid density, user i may have not enough competitive capability to become a winer. In this case, as a loser, her utility is 0. Therefore, the optimal strategy for user i is to report her true bid value and request quantity.

This remark completes the proof.

Theorem 5.2. *EFM is truthful and envy-free for* single-minded *users.*

Proof. *EFM* combines two general ideas. First is *RevenueExtractor* (F, O), which has been proven to be truthful and envy-free in Lemma 5.2. Given a target revenue F, the sale price yielded by *RevenueExtractor* (F, O) is single for one unit VM, hence, the mechanism is envy-free.

The second technique for the design of *EFM* is that of using a order-independent consensus estimate. According to Lemma 5.1, Algorithm 3 (*CATAR*) can promise that the randomized function $f(G(O))$ used by it is a consensus estimate. That is, there is $f(G(O_{-i})) = f(G(O))$ for every user i. Consider the effect of user i changing her order O_i to O_i' resulting in a new order vector O' that is the same as the original O except for user i (that is, $O'_{-i} = O_{-i}$), user i hence cannot change the target revenue F by submitting any O_i' because $f(G(O'_{-i})) = f(G(O_{-i})) = f(G(O)) = F$. Furthermore, *RevenueExtractor* is truthful, therefore, user i cannot benefit from misreporting her order, and *EFM* is truthful.

This remark completes the proof.

6. Experimental results

In the section, we perform extensive simulation experiments with real workload data extracted from Google Cluster Trace [7] to investigate the performance of our proposed auction framework. To measure the performances of the proposed mechanism, we also compare our proposed mechanism with several auction mechanisms presented as follows.

Optimal Single Price Auction (*OSPA*) is presented in Definition 4.2. The maximum revenue for cloud providers can be achieved in a single-round, single-price auction by *OSPA*. Thus, *OSPA* is viewed as a benchmark in our experimental evaluation.

Vickrey–Clarke–Groves Mechanism (*VCGM*) is a mechanism where the allocation function $\Gamma(.)$ maximizes social welfare, and the payment function $\hat{p}_i$ of user i is formulated as follows:

$$\hat{p}_i(O_i, O_{-i}) = \sum_{j \in \Gamma(O_{-i})} b_j - \sum_{j \in \Gamma(O), j \neq i} b_j \tag{28}$$

where $\sum_{j \in \Gamma(O_{-i})} b_j$ is the maximum social welfare that would have been achieved without user i, and $\sum_{j \in \Gamma(O), j \neq i} b_j$ is the aggregated users' valuations except user i. *VCGM* can achieve the property of truthfulness in an auction and has been extensively employed to design auction mechanisms, such as researches [24,20,12,13,21,22,25–28].

Uniform Price Auction (*UPA*) is an envy-free auction, where the provider serves the highest bidders first by allocating the requested number of VMs until their supply is exhausted or no more orders are placed. The sale price for one unit VM is single and decided by the lowest winning bid, thus, the mechanism is envy-free but untruthful.

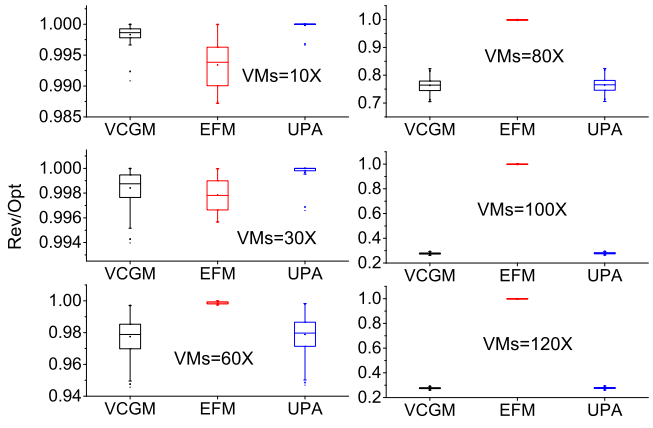
6.1. Configurations

The users' requests in our simulation experiments are extracted from Google Cluster Trace, in which the resource (*RAM*, *vCPU* and *Storage*) information is provided in normalized units. Thus, the exact number of cores or the amount of memory demanded by tasks cannot be obtained. Google Cluster traces reveal that users usually submit multiple jobs in a time interval while needing to run different types of tasks with different resource requirements simultaneously. In other words, a user may have multiple jobs, and a job is composed of one to tens of tasks, which generally execute the same binary with the same options and resource requests. Accordingly, we map each job of the user to a group of VM and her each task to one VM in the simulation experiments. Additionally, to simplify the simulations, the weighted price of each type resource of VMs is set to the same value 1; the request resources of the user with the smallest resource request amount in all the users are treated as one unit VM, and then the request quantity of other customers is scaled by it.

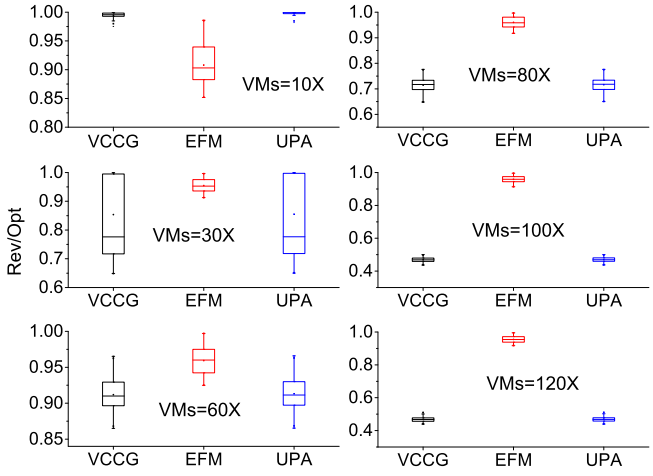
As a result, a user request contains the requested number of vCPU, the amount of memory, and the storage. The data can be collected from Google Cluster Trace (i.e., *Task Events Tables*), but the user bids have not been publicly released by any cloud providers yet. Hence, similar to [21,22], we utilize the random value that obeys a normal distribution $\mathcal{N}(0.5, 0.4)$ as the bid of the user for one unit VM. Bid price that is less than or equal to zero is discarded, and a new bid price is drawn from the distribution again. Furthermore, we also consider another user bid distribution that follows a log-normal distribution adopted in [37]. As this distribution is employed, the user bid for one unit VM is given as e^z with $z \sim \mathcal{N}(-1, 1)$ a standard normal variable in our experiments. The reserve price for one unit VM in the simulations is set to 0.12.

Moreover, the statistical results of the trace files (*Task Events Tables*) show that there are approximately 300 to 500 users who submitted their jobs to the system in an hour interval in a cell of a single cluster. The number of users in a cell is insufficient to simulate the large market. Thus, to achieve sufficient experimental data and evaluate our method for the experiments, we randomly pick 10 time periods from the trace files. Each time period lasts an hour.

To simplify experiments, assume one user in a time period submits at most one order, which consists of various types of jobs. Moreover, to prevent the customers' collusion, the permitted quantity of the resources requested by one user is up to π=300 units in our simulation experiments. Therefore, users with more than 300 unit VMs are removed, and the aggregated quantity of the effective users is rounded to 2586, and the total quantity of effective tasks approximates to 362100 unit VM in our simulation experiments.



(a) Normal distribution.



(b) Log-normal distribution.

Fig. 5. Box plots of revenue for distinct VM provisioning.

6.2. Revenue measure

An important issue is the influence of the quantity of VM provisioning on the revenue of the provider. Our simulations investigate the issue by varying the quantity of VM provisioning from 10 to 150 times the amount of the users in the cluster. The quantity of the effective users and their orders are given with the above simulation configurations. We perform the experiments to measure the revenue produced by different auction mechanisms. Each experiment is carried out 100 times to reflect the randomized nature of the auction.

As we can expect, *OSPA* produces the highest revenue in our experiments, and hence can be regarded as the benchmark in the simulations. For easy comparison, we set the evaluation metric of the revenue as the ratio of the revenue between other methods and *OSPA* scheme (*optimal revenue*). The ratio is denoted by *Rev/Opt* in our evaluation experiments.

Fig. 5 shows separate box plots of revenue generated by various VM provisioning under the simulation setting. Statistical analysis indicates that the revenue of *VCGM* and *UPA* increases first with the increase of VM provisioning, but rapidly drops while the quantity of provisioning VMs exceeds 60 times the aggregated quantity of the effective users. Whereas, the revenue of *EFM* is lower than that of the above methods when the VM provisioning is insufficient. With the increase of the VM provisioning, the revenue of *EFM* always approaches to that of *OPSA*. It implies that *EFM* is superior to above schemes in large-scale cloud markets.

Fig. 6 also presents similar results for the average revenue of the provider with same experiment configurations. The average revenue produced by *VCGM* and *UPA* is near to the optimal value when the range of the quantity of the provisioning VMs is between 10 and 60 times the total quantity of the users. Their average revenue rapidly drops while the number of VM provisioning exceeds 60 times the total quantity of the users. That is, *VCGM* and *UPA* can achieve relatively high revenue when the cloud market is under-supplied, but their performance is the worst when the cloud market is over-supplied.

Whereas, as the quantity of VM provisioning increases, the results presented in Fig. 6 show that the revenue produced by *EFM* is always close to *OSPA*, regardless of the relationship between supply and demand. The primary reason is that *EFM* allocates the VM instances to users until the total revenue is no longer improved, even though there are free resources in the system and a number of users waiting for VM allocation. In contrast, *VCGM* and *UPA* assign VM instances to users until no users request resources or no available resources are remained in the system. In doing so, a number of users with relatively lower bid become winners such that the sale price drops and results in less revenue for the cloud provider, especially during over-supplied scenario. Moreover, while the provisioning VM instances are more than 100 times, the supply of the cloud market is saturated, and the revenue generated by *VCGM* or *UPA* remains constant.

6.3. Social welfare

This subsection aims to measure the social welfare and the number of the accepted users when distinct auction mechanisms are performed. Due the winning users and the social welfare determined by *UPA* are consistent with *VCGM*, the associated details of *UPA* are omitted. In the simulations, adopting the same configurations as previous simulation experiments, we change the quantity of the provisioning VMs to compare the social welfare and the number of accepted users for different auction mechanisms. According to *VCGM*, the social welfare is formulated by

$$S(O) = \sum_{j \in \Gamma(O)} b_j \tag{29}$$

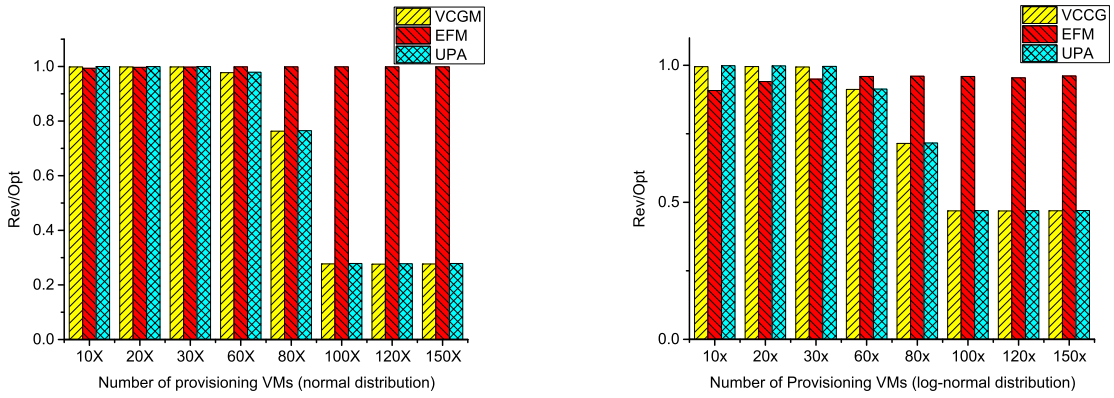
where the allocation function $\Gamma(.)$ maximizes social welfare and gives the set of winning users.

The experimental results are shown in Figs. 7 and 8. As seen in Fig. 7, the quantity of the users accepted by the mechanisms first increases with the increase of the quantity of provisioning VMs. When the provisioning VMs is more than 50 times, the quantity of accepted users by *EFM* and *OSPA* keep constant because the lower bid users are rejected for the maximization of the revenue.

As presented in Fig. 8, *EFM* can achieve the highest social welfare among these auction mechanisms in the range [10X, 40X]. Moreover, *VCGM* can obtain the highest social welfare while the number of provisioning VMs is more than 50 times, however, the revenue produced by it also rapidly decreases with the increase of VM provisioning. Generally, the cloud providers naturally aim to maximize their revenue by providing computing services for users. Hence, we argue that straightforwardly adopting *VCGM* to allocate and price cloud resources needs to be reconsidered, especially in over-supplied markets.

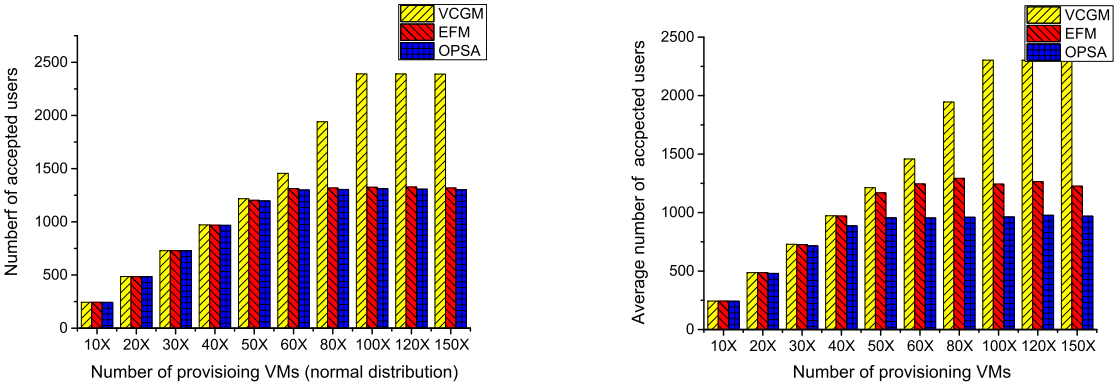
6.4. Evaluation of varying user quantity

Generally, the number of users participating in the VM competition can significantly impact the outcome of the auction-based mechanism. This subsection evaluates the performance of the mechanism by varying the quantity of the users at two levels of VM provisioning: over-supplied level and under-supplied level.



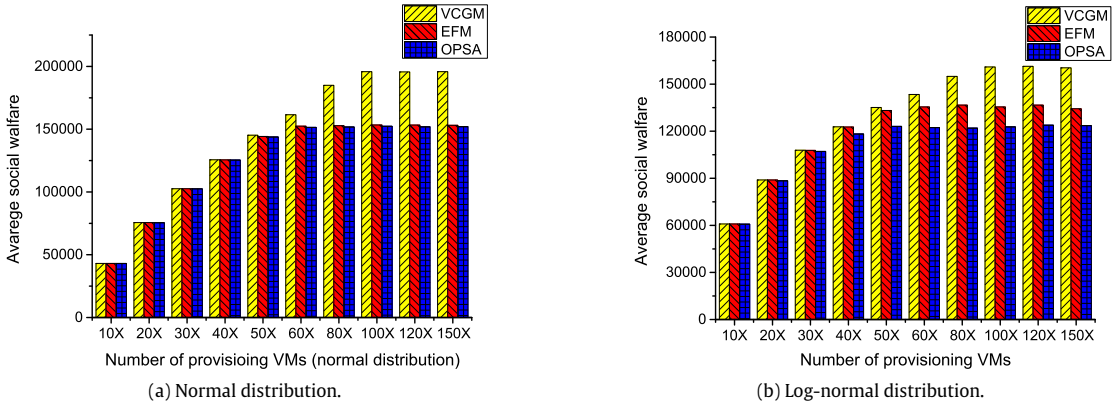
(a) Normal distribution. (b) Log-normal distribution.

Fig. 6. Average revenue.



(a) Normal distribution. (b) Log-normal distribution.

Fig. 7. Average number of served users.

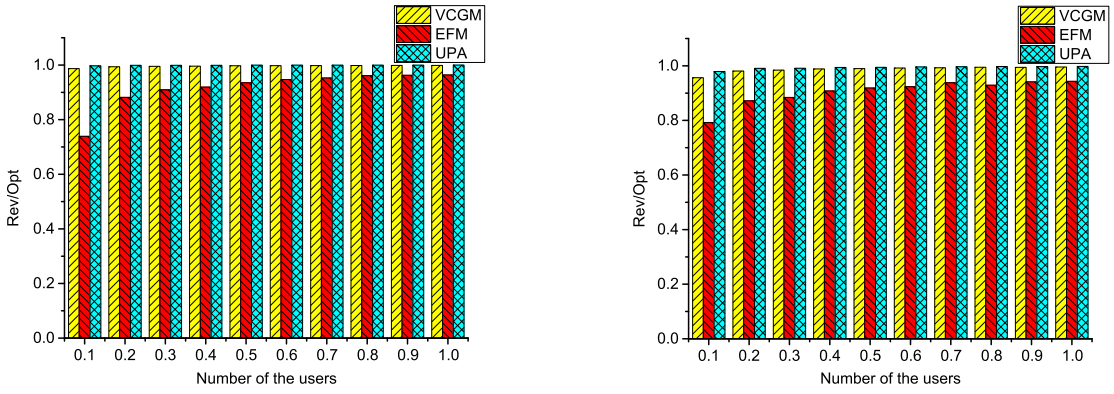


(a) Normal distribution. (b) Log-normal distribution.

Fig. 8. Average social welfare.

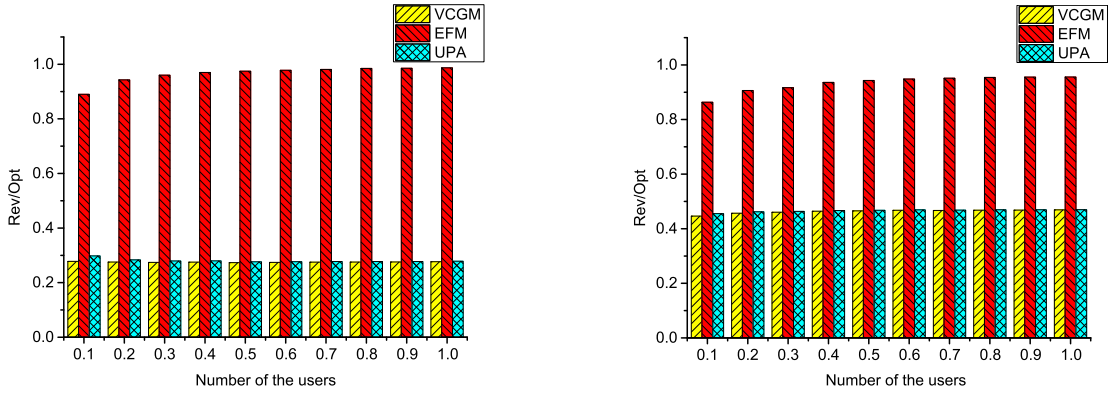
In the simulations, the quantity of the users participating in the VM competition is increased by the step size of 0.1 times the maximum number of users configured by previous experiments. In addition, following previous experiments, the state of the market is over-supplied if the quantity of provisioning VMs approaches 100times the number of the users participating in the VM competition, whereas, it will be under-supplied if the number of VM provisioning is 20 times. Accordingly, the number of provisioning VMs in the over-supplied level and under-supplied level is set to 100 and 20 times respectively.

The results presented in Figs. 9 and 10 show that the revenue of *EFM* rises with the increase of the users, and is lower than that of *VCGM* and *UPA* in the under-supplied market. By contrary, in the over-supplied market, *EFM* scheme is superior to other schemes regardless of the variation of the user quantity. Figs. 9 and 10 reveal that the quantity of the users and the relationship between supply



(a) Normal distribution. (b) Log-normal distribution.

Fig. 9. Revenue VS the number of users in the under-supplied case.



(a) Normal distribution. (b) Log-normal distribution.

Fig. 10. Revenue VS the number of users in the over-supplied case.

and demand can significantly influence the revenue produced by *EFM*. *EFM* can achieve desirable revenue being near to the optimal profit in the large-scale cloud market.

6.5. Discussion of multiple cloud providers

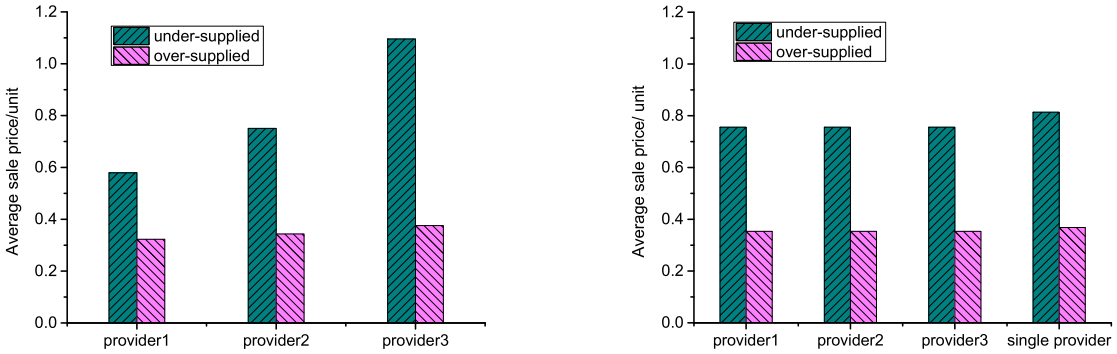
In this subsection, we will discuss the performance of our method in the scenario of multiple cloud providers. Assume that there are three cloud providers: provider1, provider2 and provider3, who have the same service capacity and offer the same products for users. The providers conduct auction with our proposed method. In the initial stage, all users are randomly allocated to providers according to the proportion of 20%, 30%, and 50%, e.g., the number of users allocated to provider1 is the product of 0.2 and the total number of users. We evaluate their average sale price per unit at two levels of VM provisioning: over-supplied level and under-supplied level. Associated experimental parameters are set in terms of the previous configurations. Each experiment is carried out 100 times to react the randomized nature of the auction.

As seen in Fig. 11(a), whether the market is over-supplied or under-supplied, the average sale price per unit of provider3 with the highest number of users is the highest, and provider1 has the lowest sale price due to the smallest number of users. Without loss of generality, because of the existence of price differences, some users will inevitably move to the providers with lower sale price. Accordingly, when the market reaches a stable state, the number of users assigned to all providers will be almost no difference. Fig. 11(b) shows the price comparison between only one provider in the market and three providers in the stable market. We can see that the sale price of the three providers is identical and close to that of only one provider in the market. It demonstrates the effectiveness of our approach under multiple providers. Nevertheless, due to the complexity of the market is far beyond the experimental environment, this issue still needs further study.

7. Conclusion

In this paper, we investigated the problem of VM allocation and pricing in the presence of multiple types of resources in clouds. We resort to the consensus auction and cost sharing mechanism to present an auction-based mechanism that is truthful, yields a close optimal profit for the cloud provider, and achieves envy-freeness through the use of a uniform price that ensures the stability of the proposed mechanism. The proposed mechanism in the over-supplied market is far superior to other schemes, such as the mechanisms based on VCG. Moreover, the mechanism has relatively low complexity of computation comparing with VCG, and thus, can be easily implemented in the large-scale cloud market. The results of simulation experiments demonstrated that our proposed mechanism can achieve desirable revenue for the cloud provider in the mass market while promising the properties of truthfulness and envy-freeness. Besides, our proposed mechanism can be applied to other resource allocation and pricing models.

In the future, we plan to explore the VM allocation and pricing among multi cloud providers as an extension of our work. We are also interested in implementing a prototype allocation system in an experimental cloud computing system to further study the performance of our proposed mechanism.



(a) In the initial state.

(b) In the stable state.

Fig. 11. The average sale price per unit for providers.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 61173107, 91320103, 61672215, U1613209), National High-tech R&D Program of China (863 Program) (No. 2012AA01A301-01), the Special Project on the Integration of Industry, Education and Research of Guangdong Province, China (No. 2012A090300003), Science research project of Department Education of Hunan Province, China (No. 2016C0270) and the Science and Technology Planning Project of Guangdong Province, China (No. 2013B090700003).

References

[1] Nguyen Cong Luong, et al., Resource management in cloud networking using economic analysis and pricing models: a survey, IEEE Commun. Surv. Tutor. 19 (2) (2017) 954–1001.
[2] N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani, Algorithmic Game Theory, Cambridge University Press, 2007.
[3] Amazon EC2 Instance Types (Online), 2017. Available: http://aws.amazon.com/ec2/instance-types.
[4] Simon Parsons, J.A. Rodriguez-Aguilar, M. Klein, Auctions and bidding: a guide for computer scientists, ACM Comput. Surv. 43 (2) (2011) 1–59.
[5] D.K. Foley, Resource allocation and the public sector, Yale Econ. Essays 7 (1) (1967) 45–98.
[6] Charles Reiss, et al., Heterogeneity and dynamicity of clouds at scale: Google trace analysis, ACM Sympos. Cloud Comput. (2012) 1–13.
[7] Google Cluster Data (Online), 2017. Available: https://code.google.com/p/googleclusterdata.
[8] Hal R. Varian, Equity, envy, and efficiency, J. Econ. Theory 9 (1) (1974) 63–91.
[9] Damien Geradin, Nicolas Petit, Damien geradin nicolas petit 2006 price discrimination under ec competition law: another antitrust doctrine in search of limiting principles?, J. Competition Law Econ. 2 (3) (2006) 479–531.
[10] Xavier León, Leandro Navarro, A stackelberg game to derive the limits of energy savings for the allocation of data center resources, Future Gener. Comput. Syst. 29 (1) (2013) 74–83.
[11] B. Yang, Z. Li, S. Chen, et al., Stackelberg game approach for energy-aware resource allocation in data centers, IEEE Trans. Parallel Distrib. Syst. 27 (12) (2016) 3646–3658.
[12] S. Zaman, D. Grosu, Combinatorial auction-based allocation of virtual machine instances in clouds, J. Parallel Distrib. Comput. 73 (4) (2013) 495–508.
[13] S. Zaman, D. Grosu, A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds, IEEE Trans. Cloud Comput. 1 (2) (2013) 129–141.
[14] M. Guzek, P. Bouvry, E.G. Talbi, A survey of evolutionary computation for resource management of processing in cloud computing, IEEE Comput. Intell. Mag. 10 (2) (2015) 53–67 (review article).
[15] J. Cao, H. Kai, K. Li, A.Y. Zomaya, Optimal multiserver configuration for profit maximization in cloud computing, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1087–1096.
[16] Nancy Samaan, A novel economic sharing model in a federation of selfish cloud providers, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2014) 12–21.
[17] Adel Nadjaran Toosi, et al., An auction mechanism for cloud spot markets, ACM Trans. Auton. Adapt. Syst. 11 (1) (2016) 1–33.
[18] Michal Feldman, Kevin Lai, Li Zhang, The proportional-share allocation market for computational resources, IEEE Trans. Parallel Distrib. Syst. 20 (8) (2009) 1075–1088.
[19] V. Di Valerio, V. Cardellini, F. Lo Presti, optimal pricing and service provisioning strategies in cloud systems: A stackelberg game approach, in: Proc. IEEE Sixth Int'l Conf. Cloud Computing, 2013, pp. 115–122.
[20] H. Zhang, B. Li, H. Jiang, F. Liu, A. Vasilakos, J. Liu, A framework for truthful online auctions in cloud computing with heterogeneous user demands, in: Proc. of IEEE INFOCOM, 2013.
[21] Mahyar Movahed Nejad, Lena Mashayekhy, Daniel Grosu, Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds, IEEE Trans. Parallel Distrib. Syst. 26 (2) (2015) 594–603.
[22] Lena Mashayekhy, et al., An online mechanism for resource allocation and pricing in clouds, IEEE Trans. Comput. 65 (4) (2016) 1172–1184.
[23] Hongbing Wang, Z. Kang, L. Wang, Performance-aware cloud resource allocation via fitness-enabled auction, IEEE Trans. Parallel Distrib. Syst. 27 (4) (2015) 1160–1173.
[24] Linquan Zhang, Z. Li, C. Wu, Dynamic resource provisioning in cloud computing: A randomized auction approach, in: IEEE INFOCOM 2014 – IEEE Conference on Computer Communications IEEE, 2014, pp. 433–441.
[25] Xiaoxi Zhang, et al. A truthful (1-ε)-optimal mechanism for on-demand cloud resource provisioning, in: Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015, pp. 1053–1061.
[26] Xiaoxi Zhang, et al., Online auctions in iaas clouds: welfare and profit maximization with server costs, ACM SIGMETRICS Perform. Eval. Rev. 43 (1) (2015) 3–15.
[27] Masahiro Tanaka, Yohei Murakami, Strategy-proof pricing for cloud service composition, IEEE Trans. Cloud Comput. 4 (3) (2016) 363–375.
[28] Marian Mihailescu, Yong Meng Teo, Strategy-proof dynamic resource pricing of multiple resource types on federated clouds, in: Int. Conf. Algorithms Archit. Parallel Process., 2010, pp. 337–350.
[29] Anna Bogomolnaia, H. Moulin, A new solution to the random Assignment Problem, J. Econom. Theory 100 (2) (2001) 295–328.
[30] Venkatesan Guruswami, et al. On profit-maximizing envy-free pricing, in: Sixteenth ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January DBLP, 2005, pp. 1164–1173.
[31] Carlee Joe-Wong, et al., Multiresource allocation: fairness-efficiency tradeoffs in a unifying framework, IEEE/ACM Trans. Netw. 21 (6) (2013) 1785–1798.
[32] Andrew V. Goldberg, Jason D. Hartline, Envy-free auctions for digital goods, in: Proceedings of the 4th ACM Conference on Electronic Commerce, ACM, 2003, pp. 29–35.
[33] Andrew V. Goldberg, J.D. Hartline, Competitiveness via consensus, in: Fourteenth ACM-SIAM Symposium on Discrete Algorithms Society for Industrial and Applied Mathematics, 2003, pp. 215–222.
[34] William Ho, X. Xu, P.K. Dey, Multi-criteria decision making approaches for supplier evaluation and selection: A literature review, European J. Oper. Res. 202 (1) (2010) 16–24.
[35] Omkarprasad S. Vaidya, S. Kumar, Analytic hierarchy process: an overview of applications, European J. Oper. Res. 169 (1) (2006) 1–29.
[36] Herv Moulin, S. Shenker, Strategyproof sharing of submodular costs:budget balance versus efficiency, Econom. Theory 18 (3) (2001) 511–533.
[37] G. Vinu Prasad, Abhinandan S. Prasad, Shrisha Rao, A combinatorial auction mechanism for multiple resource procurement in cloud computing, IEEE Trans. Cloud Comput. (2016) (in press). http://dx.doi.org/10.1109/TCC.2016.2541150.

Bo Yang, received the MSc degree in Computer Science and Technology from Hunan University, China, in 2005. He is currently working toward the Ph.D. degree at Hunan University of China. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, game theory, and mobile computing.

Zhiyong Li, received the MSc degree in System Engineering from National University of Defense Technology, Changsha, China, in 1996 and Ph.D. degree in Control Theory and Control Engineering from Hunan University, Changsha, China, in 2004. Now, he is a Full Professor with Hunan University, member of China Computer Federation (CCF). His research interests include Embedded Computing System, Visual Object Tracking, Dynamic Multi-objective Evolutionary Algorithm and Tasks Scheduling Optimization in Cloud Computing. He obtained several awards from academic organizations and conferences, such as the Champion of the Future Challenge: Intelligent Vehicles and Beyond, FC'09, which was hosted by the National Natural Science Fund Committee of China in 2009.

Shilong Jiang, received the B.S. degree from the Department of Automation Control, Taiyuan Institute of Mechanical, Taiyuan, PRC, in 1991, the MSc degree from the School of Information Engineering, Central South University of Technology, Changsha, PRC, in 1994, and the Ph.D. degree from Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, in 2000. He is with director in Laboratory of Motion Control Technology, PKU-HKUST Shenzhen-HongKong Institution. His research interests include intelligent robotics, machine vision, IoT, VR/AR, sensor fusion, multifingered manipulation, and motion control.

Keqin Li, is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber–physical systems. He has published over 480 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Services Computing, IEEE Transactions on Sustainable Computing*. He is an IEEE Fellow.