Revenue Maximization with Dynamic Auctions in IaaS Cloud Markets

Wei Wang, Ben Liang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

***Abstract*—Cloud service pricing plays a pivotal role towards the success of cloud computing. Existing pricing schemes, however, either provide no service guarantees (e.g., Spot Instances in Amazon EC2), or use static on-demand pricing in which the price cannot respond quickly to market dynamics (e.g., On-demand Instances in Amazon EC2). To overcome these problems, in this paper we design *dynamic auctions* where computing instances are periodically auctioned off to accommodate user demands over time. We address the two main challenges of revenue maximization and auction truthfulness. Our design encompasses a capacity allocation scheme, which determines the amount of instances to be auctioned off in each period, as well as the underlying auction mechanisms, based on dynamic payment schemes corresponding to the proposed capacity allocations over time. We show that our design is *two-dimensionally truthful*, and it is *asymptotically optimal* when demand is sufficiently high. Furthermore, by identifying certain optimization structures, we substantially reduce the computational complexity of our solution. Extensive simulations show that our design closely tracks market changes, while generating higher revenues than on-demand pricing.**

I. Introduction

To ensure business success, cloud vendors will need to determine how IaaS instances can be optimally priced to cloud users, so that the revenue is maximized. State-of-the-practice pricing includes three types of models, i.e., *reservation*, *on-demand*, and *bid-based* pricing. Reservation pricing charges users a one-time payment to reserve an instance for months or years [1], [2]. While reservation pricing provides long-term risk-free income to cloud vendors, its upfront payment makes it less attractive to users with flexible workloads. On-demand pricing [1], [2], [3], on the other hand, sets a fixed hourly rate and charges users for the incurred instance-hours. Despite its popularity among cloud users, the static hourly rate fails to capture market dynamics, which may result in a revenue loss for a cloud vendor. To compensate for such deficiency, Amazon has recently introduced Spot Instances, a bid-based pricing model [1]. With Spot Instances, users periodically submit bids to Amazon, who in turn posts a spot price for each period. Those users who bid higher than the spot price get their requests fulfilled in that period. This bid-based pricing offers quick response to market dynamics, yet there is no service guarantee in Amazon's design: whenever the spot price goes above the bid, the allocated instances are terminated.

Due to these drawbacks in existing pricing alternatives, we are motivated to design a new pricing strategy that combines the advantages of on-demand and bid-based pricing: it should quickly respond to market dynamics, while still offering service guarantees to cloud users. In this paper, we present our design of *dynamic auctions* in cloud markets, where a sequence of auctions are periodically carried out to accommodate demands. Users bid for instances in these auctions, and every winning user is charged a *constant* usage price produced by the auction mechanism for each instance. Requested instances are then held by winning users at the same price until the users terminate them.

Dynamic auctions are essentially a price discovery process, and are more agile responding to market dynamics. Unlike EC2 Spot Instances, our design offers *guaranteed* services: A winning user enjoys a constant price over time, and its instances will never be terminated against its will. In contrast, spot users bear the risk of price fluctuations and having all running instances terminated whenever the spot price rises above their bids.

Towards an optimal design of dynamic auctions in cloud pricing, there arise two new challenges that have never been addressed. *First,* to maximize the revenue, the underlying auctions need to be jointly designed over different time periods. With guaranteed services, an instance, once auctioned off, is no longer available for future requests until it is terminated by its user. Joint auction designs allow the vendor to optimally reserve some available capacity for future high-bid requests while rejecting current low bidders. *Second,* it is important to ensure that the joint design is *truthful* in *two-dimensional* domains — each user truthfully reports her bid price and the number of requested instances. Being truthful eliminates the user's incentive for strategic behaviours that may harm the auction integrity, and offers accurate demand information that could be utilized to predict market dynamics in the future.

To address the two challenges above, we present a near-optimal capacity allocation scheme to determine how many instances are auctioned off in each period. We jointly design the underlying auction mechanisms and specify dynamic truthful payment schemes based on the proposed capacity allocations over time. Theoretical analyses show that the proposed dynamic auctions asymptotically approach the optimal revenue as the market demand becomes sufficiently high, which is naturally the case in cloud environments. For low-demand cases, we conduct extensive simulation studies and show that our design still offers a close approximation, with revenue loss less than 2%. Simulation results also indicate that our design outperforms the regular on-demand pricing, generating a 30% revenue gain.

II. Model Formulation

Suppose a cloud vendor has allocated a fixed capacity C to host a type of virtual instances (e.g., EC2 small instances). That is, at any given time, the allocated computing resources are capable of hosting up to C instances. A sequence of auctions, indexed by $t = 1, 2, \ldots$, are periodically carried out (usually hourly, as in EC2 Spot Instances) to accommodate user requests separated in time.

A. User Model

In period t, N_t exogenous users arrive, bidding for virtual instances. We consider impatient cloud users that are interruption-intolerant. That is, users submit requests whenever their demands arise, and cannot be interrupted once their instances are held[1]. Each user i, $1 \leq i \leq N_t$, wishes to rent n_i^t virtual instances and has a maximum affordable price v_i^t, known as the *reservation value*, for renting one instance in each period. When the context is clear, we will drop the time index from the notation, e.g., n_i^t is written as n_i. The values of n_i and v_i are private information known only to user i, and are distributed with joint density function $f_{n,v}$ on the support $[1, \bar{n}] \times [0, \bar{v}]$. Denote by $F_{n,v}$ the corresponding CDF of $f_{n,v}$. Both $F_{n,v}$ and $f_{n,v}$ can be learned based on historical information, provided that the number of bidders is sufficiently large [4], [5], [6], which is naturally true for cloud markets. To join the auction, each user i submits a *two-dimensional* bid (r_i, b_i), where r_i is the number of requested instances, and b_i is the maximum price that a user wants to pay. Unless explicitly stated, all prices referred to in this paper are calculated per instance per period. Note that a user may misreport by submitting $b_i \neq v_i$ (or $r_i \neq n_i$) if it believes that this is more beneficial.

At the beginning of each period t, the vendor carries out an auction mechanism $\mathcal{M}_t$ and clears the market by deciding which user's request is fulfilled and under what price. No *partial fulfilment* is accepted. A user is either rejected or has all requested r_i instances being held, which is the case in prevalent pricing schemes [1]. Let p_i be the price charged to user i. Once p_i is set, it remains constant for user i until all requested r_i instances are terminated by this user[2]. Specifically, let $l_{i,j}$ be the running time of instance j held for user i. The utility for a winning user i is defined by

$$u_i(r_i, b_i) = \begin{cases} \sum_{j=1}^{n_i} (v_i - p_i) l_{i,j} - \sum_{j=n_i+1}^{r_i} p_i l_{i,j}\,, & \text{if } r_i \geq n_i; \\ 0\,, & \text{o.w.} \end{cases} \tag{1}$$

For those rejected users, both the charged price and the utility are zero. Note that in our model, we assume that a user derives no positive utility from overbooked resources. However, the overbooking strategy may result in a lower price p_i, and increases the overall utility u_i. Therefore, if a rational user chooses to overbook, all overbooked instances would be released in the next time period to avoid unnecessary costs, i.e., $l_{i,j} = 1$ for all $j = n_i + 1, \ldots, r_i$.

The problem of every user i is to find an optimal submission (r_i^*, b_i^*) such that its utility is maximized.

B. The Problem of Dynamic Auction Design

For the cloud vendor, suppose at time t, the available resources are sufficient to host C_t virtual instances. In addition to knowing the exact number of requests in the current time slot t, we assume that the vendor may predict the demand in the near future: it knows the distributions of N_τ, the number of users in τ, for $\tau \leq T = t + w$, with w being some *prediction window*. Note that $w = 0$ if the prediction is unavailable. Denote by $V_\tau^*(C_\tau)$ the maximum expected total revenue collected from τ to T. Let $\Gamma_{\mathcal{M}_t}(Q)$ be the *overall revenue* of auctioning Q instances to accommodate the demand in time t, using mechanism $\mathcal{M}_t$, i.e., $\Gamma_{\mathcal{M}_t}(Q) = \sum_{i=1}^{N_t} \sum_{j=1}^{r_i} p_i l_{i,j}$. The cloud vendor's problem is to jointly design a sequence of auction mechanisms $\{\mathcal{M}_t\}$ to maximize its expected revenue collected over the prediction window. We formulate this problem in the following recursive form:

$$V_t^*(C_t) = \mathbf{E}\left[\max_{\mathcal{M}_t, 0 \leq Q_t \leq C_t} \left\{ \Gamma_{\mathcal{M}_t}(Q_t) + V_{t+1}^*(C_{t+1}) \right\} \right], \tag{2}$$

where Q_t is the capacity allocated to accommodate the current demand. The boundary conditions are $V_{T+1}^*(c) = 0$ for all $c = 0, 1, \ldots, C$.

We also require the designed mechanism $\{\mathcal{M}_t\}$ to possess some salient economic properties, notably *truthfulness*. That is, for every user i, no matter how the others bid, submitting the true reservation value and the needed number of virtual instances always maximizes its utility, i.e., $u_i(n_i, v_i) \geq u_i(r_i, b_i)$ for all (r_i, b_i). Being truthful eliminates the incentive for any strategic behaviours that may harm the auction integrity, and it provides accurate market information that could be utilized to predict future demands.

However, without any *a priori* information regarding the usage patterns, i.e., the distribution of $l_{i,j}$, it would be almost impossible for a vendor to solve (2). To make the analysis tractable, we assume the instance running time is i.i.d. exponential[3]. In discrete settings, this implies that $l_{i,j}$ follows the geometric distribution with p.m.f. $P(l_{i,j} = k) = q(1-q)^{k-1}$, where q is the probability that a currently running instance will be terminated by its user in the next period. Though this is a simple model to allow tractable analysis, it has been shown to give interesting insights into practical systems. We also note that such an exponential usage pattern is widely adopted in the literature [7], [8], [9].

[1]This is a valid assumption for many cloud applications where instantaneous computations are required, e.g., website hosting, online video, and online commerce. For users that are patient and interruption-tolerant, EC2 Spot Instances constitute the best design in terms of revenue maximization, as users do not care about service guarantees in this case.

[2]Here, the price is only effective for held instances. If more instances are required, a user has to rebid for additional ones.

[3]Without this assumption, the expectation operation in (2) would be taken over the entire historical and future states of the system, and little insight can be drawn from such a model.

We now analyze the value of C_{t+1}, the available capacity in $t+1$, in (2). After Q_t instances are allocated at time t, there are $C-C_t+Q_t$ instances being held. Suppose that right before $t+1$, K of them are terminated by their users. Then at the beginning of $t+1$, there are $C_{t+1}=C_t-Q_t+K$ instances being available for new requests. Since each hosted instance has a probability q to be terminated in the next period, we know K follows binomial distribution, i.e., $K \sim Bin(C - C_t + Q_t, q)$. We re-write (2) as

$$V_t^*(C_t) = \mathbf{E}\Bigg[\max_{\mathcal{M}_t, 0\le Q_t\le C_t} \big\{\Gamma_{\mathcal{M}_t}(Q_t) + \mathbf{E}_K\big[V_{t+1}^*(C_t - Q_t + K)\big]\big\}\Bigg]. \quad (3)$$

From the above, we see that an optimal design of dynamic auctions decides not only the underlying mechanisms $\{\mathcal{M}_t\}$, but also the number of instances auctioned off in each period (i.e., $\{Q_t\}$). Note that the two decisions are closely coupled with each other. On the one hand, the instance allocation scheme is determined based on the specified underlying mechanisms. On the other hand, to ensure that the allocation scheme is not affected by strategic user behaviours, the underlying mechanisms should also take into account how the instances are allocated over time.

III. Dynamic Auction Design

This section presents our design of dynamic auction. We first characterize the revenue of truthful auctions, which allows us to focus on instance allocations first, i.e., to find the optimal Q_t. After that, we present a truthful design for the underlying mechanisms, $\mathcal{M}_t$, where the payment schemes depend on the specified allocations.

A. Characterizing Revenue for Truthful Mechanisms

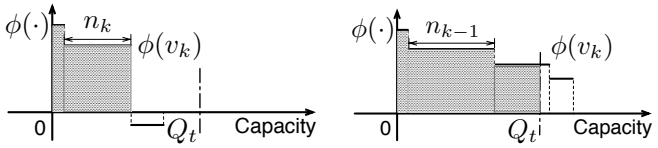
By the *Revelation Principle* [10], truthful auctions are revenue maximizing among all auctions. Hence, it suffices to focus only on truthful mechanisms when revenue is of interest. Subsequently, our discussion on capacity allocation focuses on truthful mechanisms. We therefore do not differentiate between the submitted bid (r_i, b_i) and the true request (n_i, v_i). We derive the auction revenue $\Gamma_{\mathcal{M}_t}$ as follows.

$$\mathbf{E}\left[\Gamma_{\mathcal{M}_t}\right] = \mathbf{E}\left[\sum_{i=1}^{N_t}\sum_{j=1}^{n_i} p_i l_{i,j}\right] = \frac{1}{q}\mathbf{E}\left[\sum_{i=1}^{N_t}\sum_{j=1}^{n_i} p_i\right] = \frac{1}{q}\mathbf{E}\left[\gamma_{\mathcal{M}_t}\right], \quad (4)$$

where the second equality holds because $\mathbf{E}[l_{i,j}] = 1/q$, and $\gamma_{\mathcal{M}_t}$ is the *instantaneous revenue* collected at time t.

Let $\mathbf{n} = (n_i)$ and $\mathbf{v} = (v_i)$. It has been shown in [11] that if partial fulfilment is not allowed, then the revenue of a truthful auction is characterized by

$$\mathbf{E}_{\mathbf{n},\mathbf{v}}\left[\gamma_{\mathcal{M}_t}\right] = \mathbf{E}_{\mathbf{n},\mathbf{v}}\left[\sum_{i=1}^{N_t} n_i\phi(v_i)x_i(\mathbf{n},\mathbf{v})\right], \quad (5)$$



(a) No partial fulfillment occurs. (b) Partial fulfillment occurs.

Fig. 1. The value of $\bar{\gamma}_t(Q_t)$ in two cases, shown as the shaded area. (a) The allocated capacity Q_t is sufficient to accommodate all profitable requests. (b) The allocated capacity Q_t cannot accommodate all profitable requests. This illustration shows the case where the last request is partially fulfilled.

where $\phi(v_i) = v_i - \frac{1-F_v(v_i|n_i)}{f_v(v_i|n_i)}$, and $x_i(\mathbf{n},\mathbf{v})$ takes the value 0 or 1 depending on whether user i loses or wins, respectively.

For mathematical convenience, we take a standard *regularity assumption* [12], that $\phi(\cdot)$ is increasing. This is not a restrictive assumption as it generally holds for most distributions [10].

By (5), since partial fulfilment is not supported in the cloud market [1], the expected revenue of an auction mechanism only depends on who is to win (i.e., x_i's), not what they pay (i.e., p_i). Therefore, it is equivalent to write $\gamma_{\mathcal{M}_t} = \sum_{i=1}^{N_t} n_i\phi(v_i)x_i$, and designing an optimal $\mathcal{M}_t$ is equivalent to finding a set of x_i's to maximize $\gamma_{\mathcal{M}_t}$, under the constraint that the accommodated requests do not exceed the allocated capacity. Formally, an optimal design solves the following 0-1 knapsack problem:

$$\begin{aligned} \max_{x_i\in\{0,1\}} \quad & \textstyle\sum_{i=1}^{N_t} n_i\phi(v_i)x_i \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{N_t} n_i x_i \le Q_t, \end{aligned} \quad (6)$$

where Q_t is the capacity allocated to period t.

B. Dynamic Capacity Allocation

We are now ready to derive the optimal dynamic capacity allocations. Let $\gamma_t^*(Q_t)$ be the instantaneous revenue obtained from the optimal mechanism. That is, $\gamma_t^*(Q_t) = \sum_{i=1}^{N_t} n_i\phi(v_i)x_i^*$ where x_i^* solves the knapsack problem (6). The optimal dynamic auction design problem (3) can now be equivalently written as

$$V_t^*(C_t) = \mathbf{E}\Bigg[\max_{0\le Q_t\le C_t}\big\{\frac{1}{q}\gamma_t^*(Q_t) + \mu_{t+1}^*(C_t - Q_t)\big\}\Bigg], \quad (7)$$

where $\mu_{t+1}^*(c) = \mathbf{E}_K\big[V_{t+1}^*(c+K)\big]$, with $K \sim Bin(C - c, q)$. The boundary conditions are $V_{T+1}^*(c) = 0$ for all $c = 0, 1, \ldots, C$.

Unfortunately, optimally solving (7) is computationally prohibitive, as calculating the optimal revenue $\gamma_t^*(Q_t)$ requires to solve the problem (6), which is NP-Hard. As such, we turn to an approximate solution.

Instead of calculating γ_t^*, we consider its upper bound $\bar{\gamma}_t$ obtained by solving a linear programming relaxation of (6) where fractional x_i's are allowed, i.e., $\bar{\gamma}_t(Q_t) = \max_{x_i\in[0,1]}\sum_{i=1}^{N_t} n_i\phi(v_i)x_i$, with $\sum_{i=1}^{N_t} n_i x_i \le Q_t$. The value of $\bar{\gamma}_t$ can be easily calculated as follows. Let bidders be sorted in a decreasing order of their bidding prices, i.e., $v_1 \ge v_2 \ge \cdots \ge v_{N_t}$. From the regularity assumption, this implies $\phi(v_1) \ge \cdots \ge \phi(v_{N_t})$. To compute $\bar{\gamma}_t(Q_t)$, we

sequentially accommodate bidders' requests to accumulate the revenue, from the top valued (bidder 1) to the bottom, until all capacity is allocated. Note that those negatively valued requests (i.e., $\phi(v_i) \leq 0$) will always be rejected. Figs. 1a and 1b illustrate this process, where the shaded areas represent the value of $\bar{\gamma}_t(Q_t)$. One can view the upper bound $\bar{\gamma}_t$ as the revenue obtained *as if* partial fulfilment were accepted.

The vendor now solves an approximate problem by replacing γ_t^* with $\bar{\gamma}_t$ in (7):

$$\bar{V}_t(C_t) = \mathbf{E}\bigg[\max_{0\leq Q_t\leq C_t}\big\{\frac{1}{q}\bar{\gamma}_t(Q_t) + \bar{\mu}_{t+1}(C_t - Q_t)\big\}\bigg]. \quad (8)$$

where $\bar{\mu}_{t+1}(c) = \mathbf{E}_K\big[\bar{V}_{t+1}(c+K)\big]$, $K \sim Bin(C-c, q)$. The boundary conditions are $\bar{V}_{T+1}(c) = 0$ for all $c = 1, 2, \ldots, C$. Let $Q_t^*(C_t)$ be the optimal solution to (8). The vendor then auctions off $Q_t^*(C_t)$ instances to accommodate the current demand.

We choose the above approximation for two reasons. *First,* it closely approaches the optimal revenue. *Second,* it can be efficiently calculated by identifying some optimization structures in (8). We postpone the revenue analysis until later in Sec. IV and focus on the structural and computational discussions in the following.

C. Structural Results and Capacity Allocation Rules

Directly computing capacity allocations in (8) is highly inefficient, especially for large vendors with enormous capacities. Since no closed-form solution exists, a standard method is to apply *numerical dynamic programming* [13], with the overall computational complexity of $O(C^3)$. However, in practice, a large vendor is usually capable of holding hundreds of thousands of instances of a certain type simultaneously. In this case, direct computation requires a huge amount of operations, which is prohibitive to complete within a short time.

Instead of direct computation, we identify the optimization structure of (8) and derive a simple *capacity allocation rule* that reduces the complexity to $O(C^2)$. Let $\nabla f(n) = f(n) - f(n-1)$ be the *backward difference* of a discrete function $f(n)$. We have the following optimization structure. The proof is given in our technical report [14].

Proposition 1: For every realization $\mathbf{n}$ and $\mathbf{v}$ at time $\tau = t, t+1, \ldots, T$, the optimal allocation is characterized by

$$Q_\tau^*(c) = \begin{cases} \max\limits_{1\leq n\leq c}\Big\{n : \frac{1}{q}\nabla\bar{\gamma}_\tau(n) > \nabla\bar{\mu}_{\tau+1}(c-n+1)\Big\}, \\ \qquad\qquad\qquad \text{if } \frac{1}{q}\nabla\bar{\gamma}_\tau(1) > \nabla\bar{\mu}_{\tau+1}(c); \\ 0, \qquad\qquad\qquad \text{o.w.} \end{cases} \quad (9)$$

Furthermore, $Q_\tau^*(c+1) - 1 \leq Q_\tau^*(c) \leq Q_\tau^*(c+1)$.

The second statement of Proposition 1 plays a key role in reducing the complexity to $O(C^2)$. It shows that previously calculated results can be leveraged as follows. For each period τ, $\bar{V}_\tau(c)$ is sequentially computed as $C_\tau = C, C-1, \ldots, 0$. Though the first computation ($C_\tau = C$) requires searching the entire solution space (i.e., $Q_\tau = 0, 1, \ldots, C$), subsequent computations are much more efficient. By Proposition 1, once $Q_\tau^*(C_\tau + 1)$ is calculated, one can quickly determine $Q_\tau^*(C_\tau)$ by choosing between two optimal candidates, $Q_\tau^*(C_\tau) = Q_\tau^*(C_\tau + 1)$ and $Q_\tau^*(C_\tau) = Q_\tau^*(C_\tau + 1) - 1$, and the one resulting in higher revenue $\bar{V}_\tau(C_\tau)$ is the optimal solution. The entire computation only takes $O(C^2)$ operations. We note that the complexity reduction from $O(C^3)$ of applying standard dynamic programming is significant, especially for large vendors capable of holding hundreds of thousands of instances.

D. Designing the Underlying Auction Mechanisms

With the capacity allocation rule derived above, we are now ready to present the underlying auction mechanisms. We have to ensure that the designed mechanism eliminates the incentive for any strategic behaviours by the cloud users.

One might think that the underlying auction design is trivial once the capacity allocation Q_t^* has been decided: just run existing single-round truthful mechanisms [15] to auction off Q_t^* instances in each period t. However, in the single-round auction [15], [10], [12], [16], the number of auctioned items is given and is fixed, while in our problem, users can manipulate the allocation Q_t^* by submitting different bids. For this reason, directly applying single-period truthful auctions fails to eliminate strategic behaviours in our problem.

To overcome the challenges above, we propose Algorithm 1 as the adopted mechanism, with C_t and Q_t^* being the number of available instances and the allocated capacity, respectively. In this algorithm, we artificially insert a *virtual bidder* into the market, who requests an infinite amount of instances at a price $\phi^{-1}(0)$. Introducing a virtual bidder has no effect on the auction result, since its request is never accepted, but it simplifies the revenue expression. Recall that bidders are assumed to be sorted in a decreasing order of prices, $b_1 \geq \cdots \geq b_{N_t+1}$. (There are $N_t + 1$ bidders including the virtual one.) For two bidders bidding the same price, the one requesting fewer instances ranks higher than the other.

Algorithm 1 The Truthful Mechanism $\mathcal{M}_t$ with Q_t^* Instances Allocated

1. Let k be the index such that $\sum_{j=1}^{k} r_j \leq Q_t^* < \sum_{j=1}^{k+1} r_j$
2. Let $s = \sum_{j=1}^{k} r_j$
3. Let $\hat{b}_s = \phi^{-1}(q\nabla\bar{\mu}_{t+1}(C_t - s + 1))$
4. Top k bidders win, each paying $p = \max\{b_{k+1}, \hat{b}_s\}$

Algorithm 1 adopts a similar design as the $(k+1)$-*price auction* (i.e., top k bidders win each paying the bid of bidder $k+1$), but with different payment rules. The basic idea is to charge every winner the *least bid* required to win. That is, bidder i wins if (1) it is among the top k bidders ($b_i > b_{k+1}$) and (2) all top k bids are high enough so that the allocated capacity is sufficient to accommodate all their requests. Note that if $b_k > \hat{b}_s$, then by Proposition 1, at least $s = \sum_{j=1}^{k} r_j$ instances are auctioned off: $\frac{1}{q}\nabla\bar{\gamma}_t(s) = \frac{1}{q}\phi(b_k) > \frac{1}{q}\phi(\hat{b}_s) = \nabla\bar{\mu}_{t+1}(C_t - s + 1)$.

It is worth mentioning that the (k+1)-price auction is *untruthful* for multi-demand bidders [15]. However, we show that our design completely eliminates the bidder's incentive for misreporting even in two-dimensional domains (v_i and n_i). The proof is given in [14].

Proposition 2: Algorithm 1 is two-dimensionally truthful, i.e., $u_i(n_i, v_i) \geq u_i(r_i, b_i)$ for all (r_i, b_i), $i = 1, 2, \ldots, N_t$.

Corollary 1: The designed dynamic auctions $\{\mathcal{M}_t\}$ are truthful in two-dimensional domains.

IV. Evaluations

In this section we seek to answer the following question: How well does our design approximate the optimal solution (7)? We evaluate the revenue performance of our design in two scenarios: (1) When market demand is high, we give theoretical analysis and show that the approximation is asymptotically optimal. (2) When market demand is low, we conduct extensive simulations and show that our design still closely approximates the optimal one.

A. Asymptotic Optimality for High-Demand Markets

We analyze the revenue performance of our design in a high-demand market, over the long run. Let $\gamma_t(\cdot)$ be the instantaneous revenue of Algorithm 1. Let $V_t(\cdot)$ be the expected revenue collected within the prediction window by running Algorithm 1, i.e.,

$$V_t(C_t) = \mathbf{E}\left[\frac{1}{q}\gamma_t(Q_t^*) + \mathbf{E}_K\left[V_{t+1}(C_t - Q_t^* + K)\right]\right], \quad (10)$$

where Q_t^* is obtained by solving (8) and $K \sim Bin(C - C_t + Q_t^*, q)$. The boundary conditions are $V_{T+1}(c) = 0$ for all $c = 0, 1, \ldots, C$. The following lemma conditionally bounds the revenue of our design. Its proof is given in [14].

Lemma 1: If $N_\tau \to \infty$ for all $\tau = t, \ldots, T$, then the approximate solution is $\frac{\alpha}{\alpha-1}$-competitive for any finite $\alpha > 1$. Specifically, for all C_τ, the following inequalities hold w.p.1:

$$V_\tau(C_\tau) \geq (1 - \frac{1}{\alpha})\bar{V}_\tau(C_\tau) \geq (1 - \frac{1}{\alpha})V_\tau^*(C_\tau)\,. \quad (11)$$

With Lemma 1, we see that our design is asymptotically optimal as the market demand is sufficiently high, which is naturally the case for large vendors.

Proposition 3: The expected revenue $V_t \to V_t^*$ w.p.1 if the user number $N_\tau \to \infty$ for all $\tau = t, \ldots, T$.

B. Revenue Performance in Low-Demand Markets

Even for a market where the demand is not high, our design still offers a close approximation to the optimal solution. We verify this point via extensive simulations.

We adopt a typical scenario for a medium-sized vendor with capacity $C = 10^4$. That is, up to 10^4 virtual instances of a certain type can be held at one time. User demands are separated to 300 time periods. In each period t, there are N_t users bidding for instances, each requesting n_i instances with price v_i. Let N_t be uniformly distributed in $(1, 300)$, i.e., $N_t \sim U(1, 300)$. Let $n_i \sim U(1, 100)$ and $v_i \sim U(0.05, 0.1)$. Note that Proposition 3 is no longer effective in this setting, since the market demand N_t is not sufficiently high. We enable short predictions and set the prediction window $w = 5$. (Other values of w lead to similar results.) Each result below has been averaged over 1000 runs.

Revenue vs. Time. We first evaluate the proposed dynamic auction by comparing its revenue against on-demand pricing with a fixed hourly rate (referred to as the fixed-pricing) in a long time span (from months to years). Since it is almost impossible to predict long-term future demands, a widely adopted method is to set a price p^* such that $p^*(1 - F_v(p^*))$ is maximized [6]. Note that F_v is the distribution of a user's reservation value, and $1 - F_v(p)$ is the probability that a user can afford the price p. In this sense, p^* maximizes the expected revenue collected from a single user. We adopt such fixed pricing as a benchmark in our evaluation.

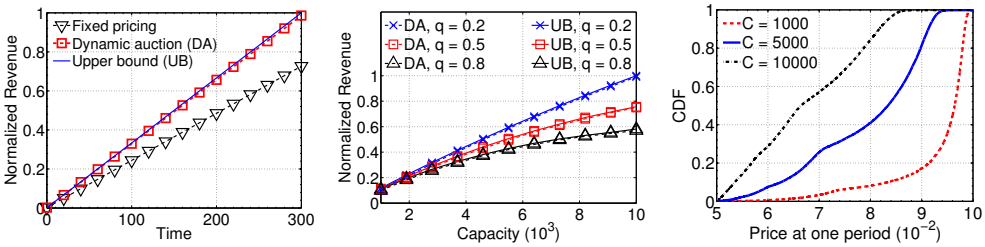
Fig. 2a illustrates the revenue performance of both dynamic auction and fixed pricing, where q is set to 0.5. The revenue is observed to be linearly increasing over time. We see that the designed mechanism accurately captures market dynamics and outperforms fixed pricing, leading to a 30% revenue improvement. Also note that our design closely approaches the theoretical revenue upper bound, with a performance gap less than 2%.

Revenue vs. Capacity. We next verify the structural results obtained in Sec. III-B by investigating the relationship between the revenue and the capacity. In particular, we conduct simulations in three different market conditions by choosing $q = 0.2, 0.5$, and 0.8, representing low, medium, and high market dynamics, respectively. For each q, we start from a low capacity setting with $C = 1000$, and increase it until $C = 10000$. For each capacity setting, we calculate the overall revenue collected in all 300 time periods. Fig. 2b illustrates the results, where the solid line represents the theoretical revenue upper bound obtained by solving (8), and the dotted line stands for the actual revenue obtained from dynamic auctions. Again, we see that the approximation design is almost optimal: in all cases, the revenue gap between the upper bound and our design is less than 2%. Also, the gap is found decreasing when more capacity is available in the datacenter. These results suggest that our design is preferred by large vendors.

Capacity vs. Price Dynamics. Our final observation is that the price becomes more dynamic as capacity increases. Fig. 2c shows the CDF of the determined prices in each time period in all 1000 runs during the simulation, where q is set to 0.5. We see that with the same demand level, a low capacity ($C = 1000$) intensifies the bid competition among users. In this case, only those who bid very high win and have their requests fulfilled. As a result, over 80% of the prices are higher than 0.09. The situation changes when more capacity is available. As supplies become more abundant, the intensity of bid competition decreases, resulting in more dynamic clearing prices (see $C = 5000$ and $C = 10000$).

V. Related Work

Many recent works in the literature advocate the use of auction-based mechanisms to allocate cloud resources to max-



(a) Normalized revenue vs. time. (b) Normalized revenue vs. capacity. (c) CDF of calculated prices.

Fig. 2. Performance evaluation of dynamic auctions (DA) and the revenue upper bounds (UB). All results are averaged over 1000 runs.

imize the revenue of a cloud vendor. Zaman and Grosu [17] propose two combinatorial auctions to allocate VMs to users. Zhang *et al.* [18] investigate a VM allocation problem for spot markets by solving a static optimization problem, yet without considering any strategic behaviours among users. Danak and Mannor [19] present two auction-based resource allocation protocols with supply adjustments. Wang *et al.* [20] also propose two combinatorial auctions to price cloud resources, and show that their designs are truthful and collusion-free. All these works focus on one-time resource allocations and restrict their discussions in a single auction period. In contrast, we consider dynamic auctions with emphasis on joint mechanism design over different time periods.

Dynamic auction design has also been studied in the economics literature. For example, Vulcano *et al.* [21] optimally design a sequence of periodic auctions to sell a certain amount of products to unit-demand buyers distributed over time. However, both optimal and truthful results obtained in [21] require the unit-demand assumption. Besides, discussions in [21] are based on the *sales market*, where products are sold to customers and are never available to sellers after a transaction. Other works that focus on revenue maximization in the sales market include [22], [23]. Unlike existing works, cloud instances are *leased*, not sold, to users: after being terminated by previous users, the instances can be hosted again to accommodate new requests. In this case, the mechanism design problem discussed in the sales market is a special case of the problem investigated in this paper.

VI. Conclusions

In this paper, we investigate into the problem of efficient pricing for revenue maximization by cloud service vendors. We advocate the use of dynamic auctions to quickly adapt to market changes while offering guaranteed services. We formulate the optimal design of dynamic auctions as an MDP, and show its NP-hardness. We turn to an approximate solution that can be directly computed within $O(C^3)$. By deriving special optimization structures of the problem, we significantly reduce the computational complexity to $O(C^2)$. A sequence of two-dimensionally truthful mechanisms are then jointly designed based on the proposed capacity allocation scheme. The construction of such mechanisms demonstrates the practical validity of the proposed revenue model. Theoretical analysis shows that our design asymptotically approaches the optimal revenue as the market demand becomes sufficiently high, which is naturally the case in reality. We show via simulations that that even for cases where the market demand is not high, our design still generates near-optimal revenue.

References

[1] Amazon EC2 Pricing, http://aws.amazon.com/ec2/pricing/.

[2] GoGrid Cloud Hosting, http://www.gogrid.com.

[3] RackSpace Cloud Hosting, http://www.rackspace.com/cloud.

[4] M. Balcan, A. Blum, J. Hartline, and Y. Mansour, "Mechanism Design via Machine Learning," in *Proc. 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005, pp. 605–614.

[5] V. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[6] I. Segal, "Optimal Pricing Mechanisms with Unknown Demand," *The American Economic Review*, vol. 93, no. 3, pp. 509–529, 2003.

[7] R. Randhawa and S. Kumar, "Usage restriction and subscription services: Operational benefits with rational users," *Manufacturing & Service Operations Management*, vol. 10, no. 3, pp. 429–447, 2008.

[8] H. Fujiwara and K. Iwama, "Average-case competitive analyses for ski-rental problems," *Algorithmica*, vol. 42, no. 1, pp. 95–107, 2005.

[9] Y. Xu and W. Xu, "Competitive algorithms for online leasing problem in probabilistic environments," in *Advances in Neural Networks (ISNN)*, 2004.

[10] R. B. Myerson, "Optimal auction design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.

[11] W. Wang, B. Li, and B. Liang, "Towards optimal capacity segmentation with hybrid cloud pricing," in *Proc. IEEE ICDCS*, 2012.

[12] J. Bulow and J. Roberts, "The simple economics of optimal auctions," *Journal of Political Economy*, vol. 97, no. 5, pp. 1060–1090, 1989.

[13] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[14] W. Wang, B. Liang, and B. Li, "Revenue Maximization with Dynamic Auctions in Cloud Markets," University of Toronto, Tech. Rep., 2013. [Online]. Available: http://iqua.ece.toronto.edu/~bli/papers/dyauc.pdf

[15] P. Klemperer, *Auctions: Theory and Practice*. Princeton University Press, 2004.

[16] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[17] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," in *Proc. IEEE CloudCom*, 2010.

[18] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," in *Proc. HOT-ICE*, 2011.

[19] A. Danak and S. Mannor, "Resource allocation with supply adjustment in distributed computing systems," in *Proc. IEEE ICDCS*, 2010.

[20] Q. Wang, K. Ren, and X. Meng, "When cloud meets eBay: Towards effective pricing for cloud computing," in *Proc. IEEE INFOCOM*, 2012.

[21] G. Vulcano, G. Van Ryzin, and C. Maglaras, "Optimal dynamic auctions for revenue management," *Management Science*, vol. 48, no. 11, pp. 1388–1407, 2002.

[22] A. Gershkov and B. Moldovanu, "Dynamic revenue maximization with heterogeneous objects: A mechanism design approach," *American Economic Journal: Microeconomics*, vol. 1, no. 2, pp. 168–198, 2009.

[23] M. Said, "Auctions with dynamic populations: Efficiency and revenue maximization," in *Proc. AMMA*, 2009.